Combinatorial approaches to RNA folding
Part III: Stocastic algorithms via language theory

Matthew Macauley

Department of Mathematical Sciences
Clemson University
`http://www.math.clemson.edu/~macaule/`

Math 4500, Fall 2016

Overview

Overview

Main question

Given a raw sequence of RNA, can we predict how it will fold?

Overview

Main question

Given a raw sequence of RNA, can we predict how it will fold?

There are two main approaches to this problem:

1. Energy minimization. Calculate the "free energy" of a folded structure. The "most likely" structures tend to be those where free energy is minimized.

   The free energy is computed recursively using dynamic programming.

2. Formal language theory. Use a formal grammar to algorithmically generate secondary structures: production rules convert symbols into strings according to the langauge's syntax.

   If we assign probabilities to the rules, then the "most likely" structure is the one that ocurrs with the highest probability.

Overview

Main question

Given a raw sequence of RNA, can we predict how it will fold?

There are two main approaches to this problem:

1. Energy minimization. Calculate the "free energy" of a folded structure. The "most likely" structures tend to be those where free energy is minimized.

   The free energy is computed recursively using dynamic programming.

2. Formal language theory. Use a formal grammar to algorithmically generate secondary structures: production rules convert symbols into strings according to the langauge's syntax.

   If we assign probabilities to the rules, then the "most likely" structure is the one that ocurrs with the highest probability.

In this lecture, we will study the formal language theory approach.

Some history

In his famous 1859 book *Evolution of the Species*, Charles Darwin wrote:

> "*the formation of different languages and of distinct species, and the proofs that both have been developed through a gradual process, are curiously parallel.*"

Some history

In his famous 1859 book *Evolution of the Species*, Charles Darwin wrote:

> "*the formation of different languages and of distinct species, and the proofs that both have been developed through a gradual process, are curiously parallel.*"

Decades later, scientists would discover a macromolecule called DNA that encoded genetic instructions for life in a mysterious language over the alphabet $\Sigma = \{a, c, g, t\}$.

Some history

In his famous 1859 book *Evolution of the Species*, Charles Darwin wrote:

> "*the formation of different languages and of distinct species, and the proofs that both have been developed through a gradual process, are curiously parallel.*"

Decades later, scientists would discover a macromolecule called DNA that encoded genetic instructions for life in a mysterious language over the alphabet $\Sigma = \{a, c, g, t\}$.

Though this would eventually lead to the fields of molecular biology and linguistics becoming interwined, major developments were needed in both fields before this could happen.

Some history

In his famous 1859 book *Evolution of the Species*, Charles Darwin wrote:

> "*the formation of different languages and of distinct species, and the proofs that both have been developed through a gradual process, are curiously parallel.*"

Decades later, scientists would discover a macromolecule called DNA that encoded genetic instructions for life in a mysterious language over the alphabet $\Sigma = \{a, c, g, t\}$.

Though this would eventually lead to the fields of molecular biology and linguistics becoming interwined, major developments were needed in both fields before this could happen.

Noam Chomsky is considered to be the father of modern linguistics. In the 1950s, he helped popularize the universal grammar theory.

Some history

In his famous 1859 book *Evolution of the Species*, Charles Darwin wrote:

> "*the formation of different languages and of distinct species, and the proofs that both have been developed through a gradual process, are curiously parallel.*"

Decades later, scientists would discover a macromolecule called DNA that encoded genetic instructions for life in a mysterious language over the alphabet $\Sigma = \{a, c, g, t\}$.

Though this would eventually lead to the fields of molecular biology and linguistics becoming interwined, major developments were needed in both fields before this could happen.

Noam Chomsky is considered to be the father of modern linguistics. In the 1950s, he helped popularize the universal grammar theory.

Chomsky's work led to a more rigorous mathematical treatment of formal langauges, revolutionizing the field of linguistics.

Some history

In his famous 1859 book *Evolution of the Species*, Charles Darwin wrote:

> "*the formation of different languages and of distinct species, and the proofs that both have been developed through a gradual process, are curiously parallel.*"

Decades later, scientists would discover a macromolecule called DNA that encoded genetic instructions for life in a mysterious language over the alphabet $\Sigma = \{a, c, g, t\}$.

Though this would eventually lead to the fields of molecular biology and linguistics becoming interwined, major developments were needed in both fields before this could happen.

Noam Chomsky is considered to be the father of modern linguistics. In the 1950s, he helped popularize the universal grammar theory.

Chomsky's work led to a more rigorous mathematical treatment of formal langauges, revolutionizing the field of linguistics.

Also in the 1950s, the structure of DNA, the newly discovered fundamental building block of life, was finally understood.

Some history

Formal langauages involve an alphabet Σ and production rules that turn symbols into substrings to generate words.

Some history

Formal langauages involve an alphabet Σ and production rules that turn symbols into substrings to generate words.

The use of formal language theory to study molecular biology began in the 1980s.

Some history

Formal langauages involve an alphabet Σ and production rules that turn symbols into substrings to generate words.

The use of formal language theory to study molecular biology began in the 1980s.

The earliest work involved using regular grammars to model biological sequences.

Some history

Formal langauages involve an alphabet Σ and production rules that turn symbols into substrings to generate words.

The use of formal language theory to study molecular biology began in the 1980s.

The earliest work involved using regular grammars to model biological sequences.

Assigning probabilities to the production rules yields *hidden Markov models* (HMMs), and these have been widely used in sequence analysis.

Some history

Formal langauages involve an alphabet Σ and production rules that turn symbols into substrings to generate words.

The use of formal language theory to study molecular biology began in the 1980s.

The earliest work involved using regular grammars to model biological sequences.

Assigning probabilities to the production rules yields *hidden Markov models* (HMMs), and these have been widely used in sequence analysis.

The location of bases in DNA and RNA strands are not uncorrelated. Regular grammars cannot model this.

Some history

Formal langauages involve an alphabet Σ and production rules that turn symbols into substrings to generate words.

The use of formal language theory to study molecular biology began in the 1980s.

The earliest work involved using regular grammars to model biological sequences.

Assigning probabilities to the production rules yields *hidden Markov models* (HMMs), and these have been widely used in sequence analysis.

The location of bases in DNA and RNA strands are not uncorrelated. Regular grammars cannot model this.

A larger class of grammars needs to be used to account for this: context-free grammars (CFGs).

Some history

Formal langauages involve an alphabet Σ and production rules that turn symbols into substrings to generate words.

The use of formal language theory to study molecular biology began in the 1980s.

The earliest work involved using regular grammars to model biological sequences.

Assigning probabilities to the production rules yields *hidden Markov models* (HMMs), and these have been widely used in sequence analysis.

The location of bases in DNA and RNA strands are not uncorrelated. Regular grammars cannot model this.

A larger class of grammars needs to be used to account for this: context-free grammars (CFGs).

Assiging probabilities to the production rules defines stochastic context-free grammars (SCFG).

What is a grammar?

What is a grammar?

Definitions

A language is a set of finite strings over an alphabet Σ of "terminal symbols".

What is a grammar?

Definitions

A language is a set of finite strings over an alphabet Σ of "terminal symbols".

A grammar is a collection of production rules that dictate how to change temporary nonterminal symbols into strings.

What is a grammar?

Definitions

A language is a set of finite strings over an alphabet Σ of "terminal symbols".

A grammar is a collection of production rules that dictate how to change temporary nonterminal symbols into strings.

One begins with a (nonterminal) start symbol S, and nonterminal symbols are repeatedly turned into strings until there are no nonterminals remaining.

What is a grammar?

Definitions

A language is a set of finite strings over an alphabet Σ of "terminal symbols".

A grammar is a collection of production rules that dictate how to change temporary nonterminal symbols into strings.

One begins with a (nonterminal) start symbol S, and nonterminal symbols are repeatedly turned into strings until there are no nonterminals remaining.

The language L generated by such a grammar is the set of all strings over Γ that can be generated in a finite number of steps from the start symbol S.

Notational convention

We will use

1. capital letters to denote nonterminal (temporary) symbols;
2. lower-case letters to denote terminal symbols;
3. greek-letters to denote strings of symbols.

What is a grammar?

An example

Consider the alphabet of terminal symbols $\Sigma = \{a, b\}$ and nonterminal symbols $N = \{S, A\}$ with production rules:

$$\begin{array}{rcl} S & \longrightarrow & aAa \\ A & \longrightarrow & bbA \mid bb \end{array}$$

What is a grammar?

An example

Consider the alphabet of terminal symbols $\Sigma = \{a, b\}$ and nonterminal symbols $N = \{S, A\}$ with production rules:

$$\begin{array}{rcl} S & \longrightarrow & aAa \\ A & \longrightarrow & bbA \mid bb \end{array}$$

The following sequence of rules below is a derivation of the string $\alpha = abbbbbba$:

$$S$$

What is a grammar?

An example

Consider the alphabet of terminal symbols $\Sigma = \{a, b\}$ and nonterminal symbols $N = \{S, A\}$ with production rules:

$$\begin{array}{rcl} S & \longrightarrow & aAa \\ A & \longrightarrow & bbA \mid bb \end{array}$$

The following sequence of rules below is a derivation of the string $\alpha = abbbbbba$:

$$S \longrightarrow aAa$$

What is a grammar?

An example

Consider the alphabet of terminal symbols $\Sigma = \{a, b\}$ and nonterminal symbols $N = \{S, A\}$ with production rules:

$$\begin{array}{rcl} S & \longrightarrow & aAa \\ A & \longrightarrow & bbA \mid bb \end{array}$$

The following sequence of rules below is a derivation of the string $\alpha = abbbbbba$:

$$S \longrightarrow a{\color{red}A}a \longrightarrow abb{\color{red}A}a$$

What is a grammar?

An example

Consider the alphabet of terminal symbols $\Sigma = \{a, b\}$ and nonterminal symbols $N = \{S, A\}$ with production rules:

$$\begin{array}{rcl} S & \longrightarrow & aAa \\ A & \longrightarrow & bbA \mid bb \end{array}$$

The following sequence of rules below is a derivation of the string $\alpha = abbbbbba$:

$$S \longrightarrow a{\color{red}A}a \longrightarrow abb{\color{red}A}a \longrightarrow abbbb{\color{red}A}a$$

What is a grammar?

An example

Consider the alphabet of terminal symbols $\Sigma = \{a, b\}$ and nonterminal symbols $N = \{S, A\}$ with production rules:

$$\begin{array}{rcl} S & \longrightarrow & aAa \\ A & \longrightarrow & bbA \mid bb \end{array}$$

The following sequence of rules below is a derivation of the string $\alpha = abbbbbba$:

$$S \longrightarrow aAa \longrightarrow abbAa \longrightarrow abbbbAa \longrightarrow abbbbbba.$$

What is a grammar?

An example

Consider the alphabet of terminal symbols $\Sigma = \{a, b\}$ and nonterminal symbols $N = \{S, A\}$ with production rules:

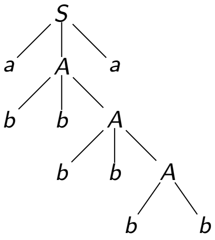
$$\begin{array}{lcl} S & \longrightarrow & aAa \\ A & \longrightarrow & bbA \mid bb \end{array}$$

The following sequence of rules below is a derivation of the string $\alpha = abbbbbba$:

$$S \longrightarrow aAa \longrightarrow abbAa \longrightarrow abbbbAa \longrightarrow abbbbbba.$$

This grammar generates the langauge precisely the set $L = \{ab^{2n}a \mid n \geq 0\}$.

The derivation shown above of the string $\alpha = abbbbbba$ can be described by the following parse tree.



Notice that α can be read off from the tree by starting at S and "walking around" the tree in a counter-clockwise order.

This grammar is context free: no terminal symbols appear on the left-hand-side of the rules.

Regular grammars

There is a hierarchy of types of grammars (The "Chomsky heirarchy"):

grammar	language	automaton	production rules
type 3	regular	finite state automata (FSA)	$A \to a,\ A \to aB$
type 2	context-free	non-deterministic pushdown automaton	$A \to \gamma$
type 1	context-sensitive	linear bounded non-deterministic Turing machine	$\alpha A \beta \to \alpha \gamma \beta$
type 0	recursiely enumerable	Turing machine	$\alpha \to \beta$

Regular grammars

There is a hierarchy of types of grammars (The "Chomsky heirarchy"):

grammar	language	automaton	production rules
type 3	regular	finite state automata (FSA)	$A \to a,\ A \to aB$
type 2	context-free	non-deterministic pushdown automaton	$A \to \gamma$
type 1	context-sensitive	linear bounded non-deterministic Turing machine	$\alpha A \beta \to \alpha\gamma\beta$
type 0	recursiely enumerable	Turing machine	$\alpha \to \beta$

If we assign probabilities to the rules, then we get stochastic versions of these grammars:

- from type 2 arises stochastic context-free grammars

Regular grammars

There is a hierarchy of types of grammars (The "Chomsky heirarchy"):

grammar	language	automaton	production rules
type 3	regular	finite state automata (FSA)	$A \to a$, $A \to aB$
type 2	context-free	non-deterministic pushdown automaton	$A \to \gamma$
type 1	context-sensitive	linear bounded non-deterministic Turing machine	$\alpha A\beta \to \alpha\gamma\beta$
type 0	recursiely enumerable	Turing machine	$\alpha \to \beta$

If we assign probabilities to the rules, then we get stochastic versions of these grammars:

- from type 2 arises stochastic context-free grammars
- from type 3 arises

Regular grammars

There is a hierarchy of types of grammars (The "Chomsky heirarchy"):

grammar	language	automaton	production rules
type 3	regular	finite state automata (FSA)	$A \to a$, $A \to aB$
type 2	context-free	non-deterministic pushdown automaton	$A \to \gamma$
type 1	context-sensitive	linear bounded non-deterministic Turing machine	$\alpha A \beta \to \alpha\gamma\beta$
type 0	recursiely enumerable	Turing machine	$\alpha \to \beta$

If we assign probabilities to the rules, then we get stochastic versions of these grammars:

- from type 2 arises stochastic context-free grammars
- from type 3 arises hidden Markov models

Stochastic context-free grammars (SCFGs)

Stochastic context-free grammars (SCFGs)

Assigning probabilities to the production rules of a CFG yields a stochastic context-free grammar (SCFG).

Stochastic context-free grammars (SCFGs)

Assigning probabilities to the production rules of a CFG yields a stochastic context-free grammar (SCFG).

In 1999, Knudsen and Hein proposed a SCFG to generate RNA secondary structures.

Stochastic context-free grammars (SCFGs)

Assigning probabilities to the production rules of a CFG yields a stochastic context-free grammar (SCFG).

In 1999, Knudsen and Hein proposed a SCFG to generate RNA secondary structures.

It can also be used to predict RNA folding, and it has comparable results to the DP energy minimization techniques.

Stochastic context-free grammars (SCFGs)

Assigning probabilities to the production rules of a CFG yields a stochastic context-free grammar (SCFG).

In 1999, Knudsen and Hein proposed a SCFG to generate RNA secondary structures.

It can also be used to predict RNA folding, and it has comparable results to the DP energy minimization techniques.

The Knudsen-Hein grammar has been implemented in the RNA folding program Pfold.

Stochastic context-free grammars (SCFGs)

Assigning probabilities to the production rules of a CFG yields a stochastic context-free grammar (SCFG).

In 1999, Knudsen and Hein proposed a SCFG to generate RNA secondary structures.

It can also be used to predict RNA folding, and it has comparable results to the DP energy minimization techniques.

The Knudsen-Hein grammar has been implemented in the RNA folding program Pfold.

Knudsen-Hein grammar

- nonterminal symbols: $\{S, L, F\}$
- terminal symbols: $\{d, d', s\}$.

The s denotes an isolated base and (d, d') denotes a base pair.

Production rules:

$$\begin{array}{llcll}
S \longrightarrow LS & \text{with probability } p_1 & \text{or} & L & \text{with probability } q_1 \\
L \longrightarrow dFd' & \text{with probability } p_2 & \text{or} & s & \text{with probability } q_2 \\
F \longrightarrow dFd' & \text{with probability } p_3 & \text{or} & LS & \text{with probability } q_3.
\end{array}$$

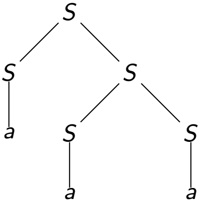
An ambiguous grammar

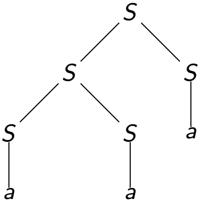
Consider the grammar $S \longrightarrow SS|a$.

An ambiguous grammar

Consider the grammar $S \longrightarrow SS|a$.

There are multiple leftmost derivations of the string aaa. Here are two possible left parse trees:





$S \rightarrow SS \rightarrow aS \rightarrow aSS \rightarrow aaS \rightarrow aaa$

$S \rightarrow SS \rightarrow SSS \rightarrow aSS \rightarrow aaS \rightarrow aaa$

The Knudsen-Hein grammar

Production rules:

$$\begin{array}{llcll}
S \longrightarrow LS & \text{with probability } p_1 & \text{or} & L & \text{with probability } q_1 \\
L \longrightarrow dFd' & \text{with probability } p_2 & \text{or} & s & \text{with probability } q_2 \\
F \longrightarrow dFd' & \text{with probability } p_3 & \text{or} & LS & \text{with probability } q_3.
\end{array}$$

Comments

The Knudsen-Hein grammar

Production rules:

$$\begin{array}{llcll} S \longrightarrow LS & \text{with probability } p_1 & \text{or} & L & \text{with probability } q_1 \\ L \longrightarrow dFd' & \text{with probability } p_2 & \text{or} & s & \text{with probability } q_2 \\ F \longrightarrow dFd' & \text{with probability } p_3 & \text{or} & LS & \text{with probability } q_3. \end{array}$$

Comments

- the start symbol S produces loops

The Knudsen-Hein grammar

Production rules:

$$\begin{array}{llcll} S \longrightarrow LS & \text{with probability } p_1 & \text{or} & L & \text{with probability } q_1 \\ L \longrightarrow dFd' & \text{with probability } p_2 & \text{or} & s & \text{with probability } q_2 \\ F \longrightarrow dFd' & \text{with probability } p_3 & \text{or} & LS & \text{with probability } q_3. \end{array}$$

Comments

- the start symbol S produces loops
- the nonterminal F makes stacks

The Knudsen-Hein grammar

Production rules:

$$\begin{array}{llcll}
S \longrightarrow LS & \text{with probability } p_1 & \text{or} & L & \text{with probability } q_1 \\
L \longrightarrow dFd' & \text{with probability } p_2 & \text{or} & s & \text{with probability } q_2 \\
F \longrightarrow dFd' & \text{with probability } p_3 & \text{or} & LS & \text{with probability } q_3.
\end{array}$$

Comments

- the start symbol S produces loops
- the nonterminal F makes stacks
- the rule $F \to LS$ ensures that each hairpin loop has size ≥ 3. at least 3.

The Knudsen-Hein grammar

Production rules:

$$\begin{array}{llcll} S \longrightarrow LS & \text{with probability } p_1 & \text{or} & L & \text{with probability } q_1 \\ L \longrightarrow dFd' & \text{with probability } p_2 & \text{or} & s & \text{with probability } q_2 \\ F \longrightarrow dFd' & \text{with probability } p_3 & \text{or} & LS & \text{with probability } q_3. \end{array}$$

Comments

- the start symbol S produces loops
- the nonterminal F makes stacks
- the rule $F \to LS$ ensures that each hairpin loop has size ≥ 3. at least 3.
- if one wanted hairpin loops to have size ≥ 4, this would have to be changed to $F \to LLS$.

The Knudsen-Hein grammar

Production rules:

$$\begin{array}{llcll}
S \longrightarrow LS & \text{with probability } p_1 & \text{or} & L & \text{with probability } q_1 \\
L \longrightarrow dFd' & \text{with probability } p_2 & \text{or} & s & \text{with probability } q_2 \\
F \longrightarrow dFd' & \text{with probability } p_3 & \text{or} & LS & \text{with probability } q_3.
\end{array}$$

Comments

- the start symbol S produces loops
- the nonterminal F makes stacks
- the rule $F \to LS$ ensures that each hairpin loop has size ≥ 3. at least 3.
- if one wanted hairpin loops to have size ≥ 4, this would have to be changed to $F \to LLS$.
- Since the p_i's and q_i's are probabilities, they must satisfy $p_i + q_i = 1$.

The Knudsen-Hein grammar

Production rules:

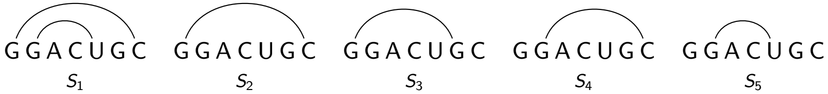
$$\begin{array}{llcll} S \longrightarrow LS & \text{with probability } p_1 & \text{or} & L & \text{with probability } q_1 \\ L \longrightarrow dFd' & \text{with probability } p_2 & \text{or} & s & \text{with probability } q_2 \\ F \longrightarrow dFd' & \text{with probability } p_3 & \text{or} & LS & \text{with probability } q_3. \end{array}$$

Comments

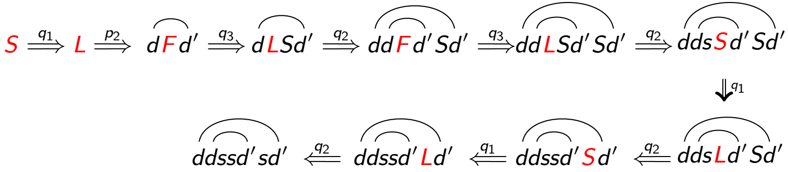
- the start symbol S produces loops
- the nonterminal F makes stacks
- the rule $F \to LS$ ensures that each hairpin loop has size ≥ 3. at least 3.
- if one wanted hairpin loops to have size ≥ 4, this would have to be changed to $F \to LLS$.
- Since the p_i's and q_i's are probabilities, they must satisfy $p_i + q_i = 1$.
- This grammar is unambiguous.

The Knudsen-Hein grammar: an example

Consider the sequence $\mathbf{b} =$ G G A C U G C, which can fold into seven secondary structures, if one allows loop sizes of minimum length 3. In addition to the unfolded structure S_0, here are five of the six others:



Here is the derivation of the first secondary structure shown above:



$$S \xRightarrow{q_1} L \xRightarrow{p_2} dFd' \xRightarrow{q_3} dLSd' \xRightarrow{q_2} ddFd'Sd' \xRightarrow{q_3} ddLSd'Sd' \xRightarrow{q_2} ddsSd'Sd'$$

$$\Downarrow^{q_1}$$

$$ddssd'sd' \xLeftarrow{q_2} ddssd'Ld' \xLeftarrow{q_1} ddssd'Sd' \xLeftarrow{q_2} ddsLd'Sd'$$

The probability of generating this secondary structure S_1 with the Knudsen-Hein grammar is

$$P(S_1) = q_1 p_2 q_3 q_2 q_3 q_2 q_1 q_2 q_1 q_2 = p_2^2 q_1^3 q_2^3 q_3^2 \,.$$

The Knudsen-Hein grammar: another example

The following is a derivation of the structure S_2 from the previous example:

$$S \xrightarrow{q_1} L \xrightarrow{p_2} \overparen{dFd'} \xrightarrow{q_3} \overparen{dLSd'} \xrightarrow{p_1^3} \overparen{dLLLSd'} \xrightarrow{q_1} \overparen{dLLLLd'} \xrightarrow{q_2^5} \overparen{dsssssd'}$$

The prediction problem

What's missing?

The prediction problem

What's missing?

- The Knudsen-Hein grammar generates structures – no bases yet!

The prediction problem

What's missing?

- The Knudsen-Hein grammar generates structures – no bases yet!
- This doesn't tell us how to predict anything.

The prediction problem

What's missing?

- The Knudsen-Hein grammar generates structures – no bases yet!
- This doesn't tell us how to predict anything.

Suppose we begin with the sequence $\mathbf{b} = GGACUGC$. As we've seen, there are six possible ways it can fold. Which is most likely?

The prediction problem

What's missing?

- The Knudsen-Hein grammar generates structures – no bases yet!
- This doesn't tell us how to predict anything.

Suppose we begin with the sequence $\mathbf{b} = GGACUGC$. As we've seen, there are six possible ways it can fold. Which is most likely?

Assuming that our sequence is $\mathbf{b}$ (this is a "conditional probability"), the probability of it folding into S_i is simply a weighted average:

$$P(S_i \mid \mathbf{b}) = \frac{P(S_i)}{P(S_0) + P(S_1) + P(S_2) + P(S_3) + P(S_4) + P(S_5)}.$$

If we knew the values of each p_i and q_i, then it would be easy to determine which structure is most likely.

The prediction problem

What's missing?

- The Knudsen-Hein grammar generates structures – no bases yet!
- This doesn't tell us how to predict anything.

Suppose we begin with the sequence $\mathbf{b} = GGACUGC$. As we've seen, there are six possible ways it can fold. Which is most likely?

Assuming that our sequence is $\mathbf{b}$ (this is a "conditional probability"), the probability of it folding into S_i is simply a weighted average:

$$P(S_i \mid \mathbf{b}) = \frac{P(S_i)}{P(S_0)+P(S_1)+P(S_2)+P(S_3)+P(S_4)+P(S_5)}.$$

If we knew the values of each p_i and q_i, then it would be easy to determine which structure is most likely.

Alternatively, if we knew the actual distribution of structures (the "weighted average"), then it would be easy to determine p_i and q_i.

The prediction problem

What's missing?

- The Knudsen-Hein grammar generates structures – no bases yet!
- This doesn't tell us how to predict anything.

Suppose we begin with the sequence $\mathbf{b} = GGACUGC$. As we've seen, there are six possible ways it can fold. Which is most likely?

Assuming that our sequence is **b** (this is a "conditional probability"), the probability of it folding into S_i is simply a weighted average:

$$P(S_i \mid \mathbf{b}) = \frac{P(S_i)}{P(S_0) + P(S_1) + P(S_2) + P(S_3) + P(S_4) + P(S_5)}.$$

If we knew the values of each p_i and q_i, then it would be easy to determine which structure is most likely.

Alternatively, if we knew the actual distribution of structures (the "weighted average"), then it would be easy to determine p_i and q_i.

Unfortunately, *a priori*, we know neither.

The prediction problem

For example, here are the probabilties of each secondary structure conditioned on the the fixed sequence $\mathbf{b} = GGACUGC$. (Since $P(S_2|\mathbf{b}) = P(S_3|\mathbf{b}) = P(S_4|\mathbf{b})$, only one of these is listed.)

(p_1, q_1)	(p_2, q_2)	(p_3, q_3)	$P(S_0|\mathbf{b})$	$P(S_1|\mathbf{b})$	$P(S_2|\mathbf{b})$	$P(S_5|\mathbf{b})$
(.45, .55)	(.5, .5)	(.5, .5)	.01142	.45772	.07583	.30335
(.5, .5)	(.5, .5)	(.5, .5)	.02222	.35556	.08889	.35556
(.25, .75)	(.75, .25)	(.25, .75)	.00000	.98017	.00227	.01211
(.75, .25)	(.25, .75)	(.75, .25)	.64390	.00279	.04240	.22612
(.75, .25)	(.75, .25)	(.75, .25)	.07511	.07913	.04451	.7122
(.869, .131)	(.788, .212)	(.895, .105)	.25314	.00568	.01547	.69477

The prediction problem

For example, here are the probabilties of each secondary structure conditioned on the the fixed sequence $\mathbf{b} = GGACUGC$. (Since $P(S_2|\mathbf{b}) = P(S_3|\mathbf{b}) = P(S_4|\mathbf{b})$, only one of these is listed.)

(p_1, q_1)	(p_2, q_2)	(p_3, q_3)	$P(S_0\|\mathbf{b})$	$P(S_1\|\mathbf{b})$	$P(S_2\|\mathbf{b})$	$P(S_5\|\mathbf{b})$
(.45, .55)	(.5, .5)	(.5, .5)	.01142	.45772	.07583	.30335
(.5, .5)	(.5, .5)	(.5, .5)	.02222	.35556	.08889	.35556
(.25, .75)	(.75, .25)	(.25, .75)	.00000	.98017	.00227	.01211
(.75, .25)	(.25, .75)	(.75, .25)	.64390	.00279	.04240	.22612
(.75, .25)	(.75, .25)	(.75, .25)	.07511	.07913	.04451	.7122
(.869, .131)	(.788, .212)	(.895, .105)	.25314	.00568	.01547	.69477

What remains to be done:

1. Find the probability parameters. Can be done by either:
   - the Cocke-Younger-Kasami (CYK) algorithm; [HMM analogue: Vitebri algorithm]
   - the inside-outside algorithm; [HMM analgoue: forward-backward algorithm]
2. Find the most likely derivation. Done by dynamic programming.