

# The (Context-Free) Grammatical Approach

The core of a context-free grammar is a set of production rules that replaces single variables with strings of variables and symbols

The grammar can generate all strings that, starting with a special start variable, can be obtained by applying the production rules until no variables remain

**Example:**  $S \rightarrow aSb \mid \epsilon$  generates the set of strings  $\{a^n b^n \mid n \geq 0\}$ ; e.g.  $aabb$  is generated as  $S \Rightarrow aSb$

# The (Context-Free) Grammatical Approach

The core of a context-free grammar is a set of production rules that replaces single variables with strings of variables and symbols

The grammar can generate all strings that, starting with a special start variable, can be obtained by applying the production rules until no variables remain

**Example:**  $S \rightarrow aSb \mid \epsilon$  generates the set of strings  $\{a^n b^n \mid n \geq 0\}$ ; e.g.  $aabb$  is generated as  $S \Rightarrow aSb \Rightarrow aaSbb$

# The (Context-Free) Grammatical Approach

The core of a context-free grammar is a set of production rules that replaces single variables with strings of variables and symbols

The grammar can generate all strings that, starting with a special start variable, can be obtained by applying the production rules until no variables remain

**Example:**  $S \rightarrow aSb \mid \epsilon$  generates the set of strings  $\{a^n b^n \mid n \geq 0\}$ ; e.g.  $aabb$  is generated as  $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$ .

# Mfold Recursions

We can now formulate recursions for computing the minimum free energy of any structure for a given sequence.

$$\begin{array}{c}
 \text{Diagram: A semi-circle with a dashed green top and a solid green bottom. The bottom is a horizontal line from point 1 to point i. The area inside is shaded gray.} \\
 wx(1, i) \\
 = \min \left\{ \text{Diagram: A semi-circle with a dashed green top and a solid green bottom. The bottom is a horizontal line from point 1 to point i. The area inside is shaded gray.}, \min_j \left\{ \text{Diagram: A semi-circle with a dashed green top and a solid blue bottom. The bottom is a horizontal line from point 1 to point i. The area inside is shaded gray.} \right\} \right\}
 \end{array}$$

$$\begin{array}{c}
 \text{Diagram: A semi-circle with a jagged blue top and a solid blue bottom. The bottom is a horizontal line from point i to point j. The area inside is shaded gray.} \\
 vx(i, j) \\
 = \min \left\{ \text{Diagram: A circle with a jagged blue top and a solid blue bottom. The bottom is a horizontal line from point i to point j. The area inside is shaded gray.}, \min_{\substack{k, l \\ (k-i) + (j-l) < c}} \left\{ \text{Diagram: A semi-circle with a jagged blue top and a solid blue bottom. The bottom is a horizontal line from point i to point j. The area inside is shaded gray.} \right\}, \min_k \left\{ \text{Diagram: A semi-circle with a jagged blue top and a solid red bottom. The bottom is a horizontal line from point i to point j. The area inside is shaded gray.} \right\} \right\}
 \end{array}$$

$$\begin{array}{c}
 \text{Diagram: A semi-circle with a dashed red top and a solid red bottom. The bottom is a horizontal line from point i to point j. The area inside is shaded gray.} \\
 wx_I(i, j) \\
 = \min \left\{ \text{Diagram: A semi-circle with a dashed red top and a solid red bottom. The bottom is a horizontal line from point i to point j. The area inside is shaded gray.}, \text{Diagram: A semi-circle with a dashed red top and a solid red bottom. The bottom is a horizontal line from point i to point j. The area inside is shaded gray.}, \text{Diagram: A semi-circle with a jagged blue top and a solid blue bottom. The bottom is a horizontal line from point i to point j. The area inside is shaded gray.}, \min_k \left\{ \text{Diagram: A semi-circle with a dashed red top and a solid blue bottom. The bottom is a horizontal line from point i to point j. The area inside is shaded gray.} \right\} \right\}
 \end{array}$$

Current energy parameters allows predictions that on average find between 56% and 70% of the base pairs in known structures.

# The (Context-Free) Grammatical Approach

The core of a context-free grammar is a set of production rules that replaces single variables with strings of variables and symbols. The grammar generates all strings that, starting with a special start variable, can be obtained by applying the production rules until no variables remain,

**Example:**  $S \rightarrow aSb \mid \epsilon$  generates the set of strings  $\{a^n b^n \mid n \geq 0\}$ ; e.g.  $aabb$  is generated as  $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$ .

From the mfold recursions it is easy to develop a grammar that captures the secondary structure of RNA sequences:

$$S \rightarrow S\sigma \mid SA \mid \epsilon$$

$$A \rightarrow \sigma L\bar{\sigma} \mid \sigma LAL\bar{\sigma} \mid \sigma BB\bar{\sigma}$$

$$B \rightarrow B\sigma \mid \sigma B \mid A \mid BA$$

$$L \rightarrow \sigma L \mid \epsilon$$

By assigning probabilities to the productions, the secondary structure prediction problem becomes the problem of finding the most probable parse

# Pros and Cons

## Advantages:

- Allows full probabilistic models of RNA secondary structures; inferring probabilities from biological structures elements are scored biologically rather than thermodynamically
- Provides a well developed formalism for specifying structural elements, as well as parsing programs

## Pitfalls:

- Doesn't allow as general a scoring scheme, e.g. for internal loops, without increasing time and/or space complexity
- Care has to be taken to make sure that there is no more than one parse corresponding to the same secondary structure – the same problem that we saw for partition function calculations

# The Full Partition Function

Our energy model defines a probability distribution on secondary structures

The partition function is the sum of Boltzmann terms from all structures

$$\begin{array}{c}
 \text{Diagram: A semi-circle with a dashed green top boundary and a solid green bottom boundary between points 1 and i.} \\
 wx(1, i)
 \end{array}
 =
 \begin{array}{c}
 \text{Diagram: A semi-circle with a dashed green top boundary and a solid green bottom boundary between points 1 and i, ending with a horizontal line to point i.} \\
 \text{Diagram: A semi-circle with a dashed green top boundary and a solid blue bottom boundary between points 1 and j, ending with a horizontal line to point i.} \\
 \sum_j
 \end{array}$$

$$\begin{array}{c}
 \text{Diagram: A semi-circle with a solid blue bottom boundary between points i and j, and a jagged blue top boundary.} \\
 vx(i, j)
 \end{array}
 =
 \begin{array}{c}
 \text{Diagram: A circle with a jagged black top boundary and a solid black bottom boundary between points i and j.} \\
 \text{Diagram: A semi-circle with a jagged black top boundary and a solid blue bottom boundary between points i and j, with an internal jagged black boundary between points k and l.} \\
 \sum_{k,l}
 \end{array}
 +
 \begin{array}{c}
 \text{Diagram: A semi-circle with a dashed red top boundary and a solid red bottom boundary between points i and j, with a jagged black top boundary extending from i to j.}
 \end{array}$$

$$\begin{array}{c}
 \text{Diagram: A semi-circle with a dashed yellow top boundary and a solid yellow bottom boundary between points i and j.} \\
 wx_1(i, j)
 \end{array}
 =
 \begin{array}{c}
 \text{Diagram: A semi-circle with a dashed yellow top boundary and a solid blue bottom boundary between points i and j, with a jagged blue top boundary between points k and l.} \\
 \sum_k
 \end{array}
 +
 \begin{array}{c}
 \text{Diagram: A semi-circle with a dashed yellow top boundary and a solid black bottom boundary between points i and j.}
 \end{array}$$

$$\begin{array}{c}
 \text{Diagram: A semi-circle with a dashed red top boundary and a solid red bottom boundary between points i and j.} \\
 wx_2(i, j)
 \end{array}
 =
 \begin{array}{c}
 \text{Diagram: A semi-circle with a dashed red top boundary and a solid red bottom boundary between points i and j, ending with a horizontal line to point j.} \\
 \text{Diagram: A semi-circle with a dashed yellow top boundary and a solid blue bottom boundary between points i and j, with a jagged blue top boundary between points k and l.} \\
 \sum_k
 \end{array}
 +
 \begin{array}{c}
 \text{Diagram: A semi-circle with a dashed red top boundary and a solid blue bottom boundary between points i and j, with a jagged blue top boundary between points i and k.} \\
 \sum_k
 \end{array}$$

# The (Context-Free) Grammatical Approach

The core of a context-free grammar is a set of production rules that replaces single variables with strings of variables and symbols. The grammar generates all strings that, starting with a special start variable, can be obtained by applying the production rules until no variables remain,

**Example:**  $S \rightarrow aSb \mid \epsilon$  generates the set of strings  $\{a^n b^n \mid n \geq 0\}$ ; e.g.  $aabb$  is generated as  $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$ .

From the mfold recursions it is easy to develop a grammar that captures the secondary structure of RNA sequences:

$$S \rightarrow S\sigma \mid SA \mid \epsilon$$

$$A \rightarrow \sigma L\bar{\sigma} \mid \sigma LAL\bar{\sigma} \mid \sigma C\bar{\sigma}$$

$$B \rightarrow LA \mid B\sigma$$

$$C \rightarrow C\sigma \mid BA \mid CA$$

$$L \rightarrow \sigma L \mid \epsilon$$

By assigning probabilities to the productions, the secondary structure prediction problem becomes the problem of finding the most probable parse

# Algorithmic Principle – Hierarchical Decomposition

RNA secondary structure prediction and (S)CFG parsing belongs to class of methods where solution is found by *hierarchically* decomposing into smaller problems

Compare to alignment where we *sequentially* decompose the problem into smaller problems via recursions of the form

$$\begin{array}{c} \bullet \text{---} \bullet \\ | \quad | \\ 1 \quad j \\ \bullet \text{---} \bullet \\ | \quad | \\ 1 \quad i \end{array} = \min \left\{ 1 + \begin{array}{c} \bullet \text{---} \bullet \\ | \quad | \\ 1 \quad j \\ \bullet \text{---} \bullet \\ | \quad | \\ 1 \quad i-1 \end{array}, 1 + \begin{array}{c} \bullet \text{---} \bullet \\ | \quad | \\ 1 \quad j-1 \\ \bullet \text{---} \bullet \\ | \quad | \\ 1 \quad i \end{array}, \delta(s[i], s[j]) + \begin{array}{c} \bullet \text{---} \bullet \\ | \quad | \\ 1 \quad j-1 \\ \bullet \text{---} \bullet \\ | \quad | \\ 1 \quad i-1 \end{array} \right\}$$

$A(i, j)$

Adding inversions would require hierarchical decomposition solution

$$\begin{array}{c} \bullet \text{---} \bullet \\ | \quad | \\ k \quad l \\ \bullet \text{---} \bullet \\ | \quad | \\ i \quad j \end{array} = \min \left\{ \begin{array}{l} \min \left\{ 1 + \begin{array}{c} \bullet \text{---} \bullet \\ | \quad | \\ k \quad l \\ \bullet \text{---} \bullet \\ | \quad | \\ i \quad j-1 \end{array}, 1 + \begin{array}{c} \bullet \text{---} \bullet \\ | \quad | \\ k \quad l-1 \\ \bullet \text{---} \bullet \\ | \quad | \\ i \quad j \end{array}, \delta(s[i], s[j]) + \begin{array}{c} \bullet \text{---} \bullet \\ | \quad | \\ k \quad l-1 \\ \bullet \text{---} \bullet \\ | \quad | \\ i \quad j-1 \end{array} \right\} \\ \min_{a,b} \left\{ 1 + \begin{array}{c} \bullet \text{---} \bullet \\ | \quad | \\ k \quad b \\ \bullet \text{---} \bullet \\ | \quad | \\ i \quad a \end{array} + \begin{array}{c} \bullet \text{---} \bullet \\ | \quad | \\ l \quad b+1 \\ \bullet \text{---} \bullet \\ | \quad | \\ a+1 \quad j \end{array} \right\} \end{array} \right\}$$

$A(i, j; k, l)$

## Non-biological Example – Matrix Multiplication

Matrices are arrays of numbers, e.g.  $\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & -1 \end{pmatrix}$

A matrix with 3 rows and 2 columns is called a  $3 \times 2$  matrix

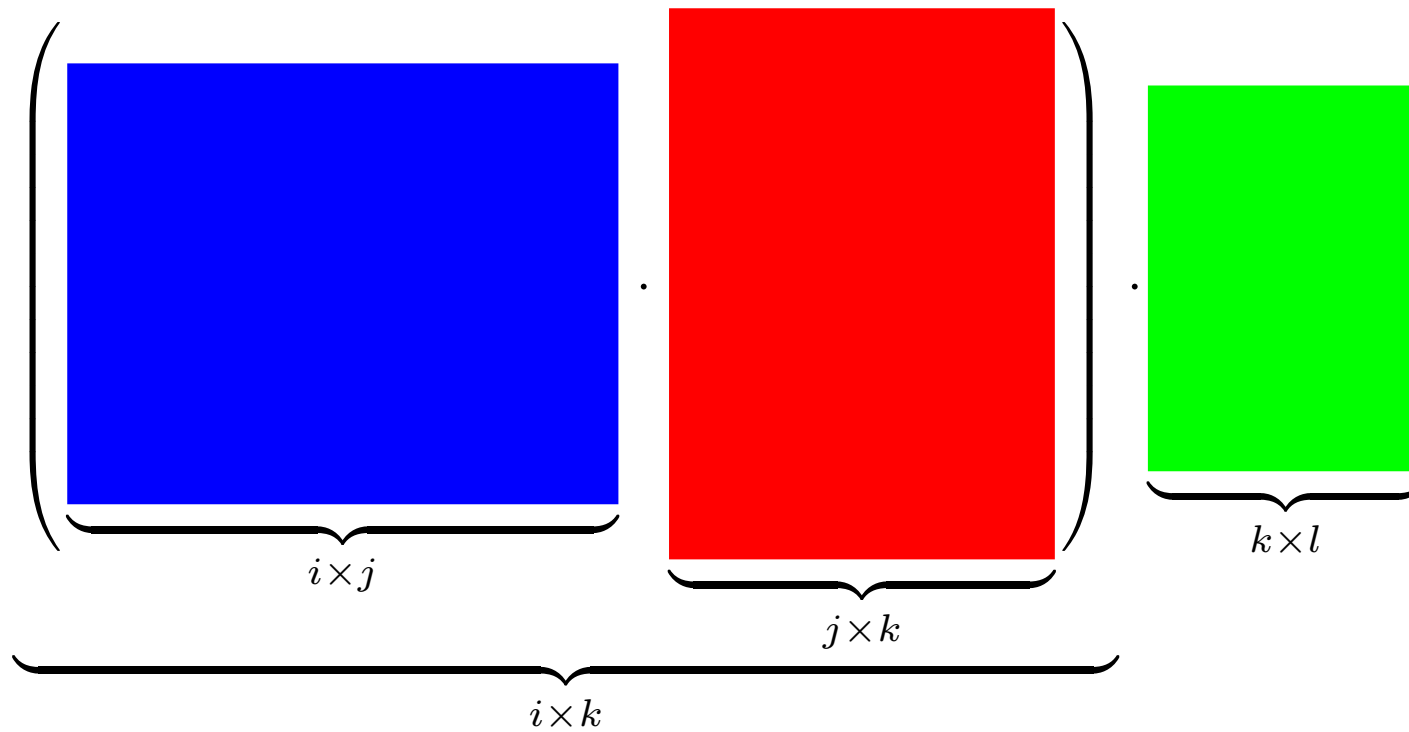
An  $i \times j$  matrix can be multiplied by a  $j \times k$  matrix to yield a  $i \times k$  matrix

The entry in the  $i$ 'th row and  $j$ 'th column of the product of matrices  $A$  and  $B$  is  $\sum_k A_{i,k} \cdot B_{k,j}$

Clearly the product of matrices of dimensions  $i \times j$  and  $j \times k$  can be computed in time  $O(ijk)$

# Non-biological Example – Matrix Chain Multiplication

Matrix multiplication is associative so a chain of matrix multiplications can be done in various orders:

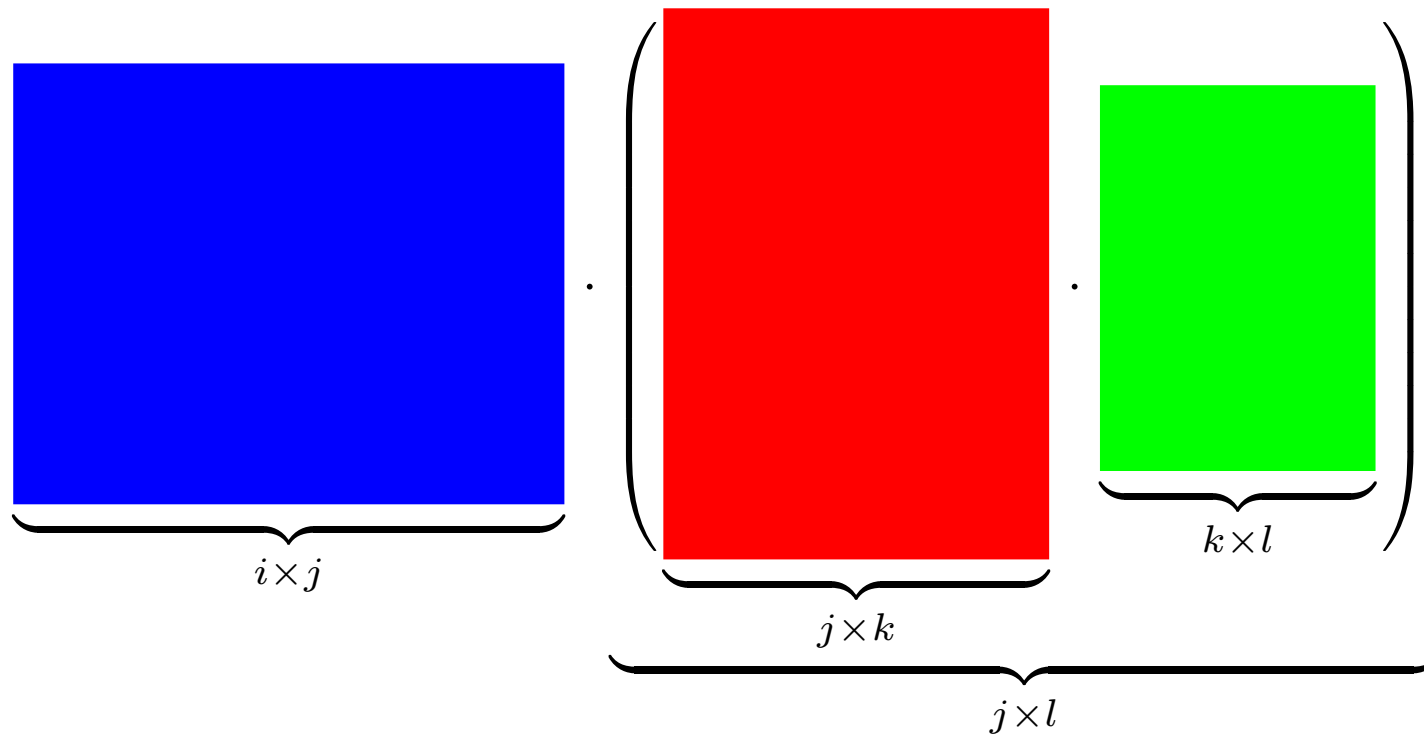


Total computation time  $O(ijk) + O(ikl) = O(ik(j + l))$

$i = k = 1, j = l = 10 \Rightarrow ik(j + l) = 20$

# Non-biological Example – Matrix Chain Multiplication

Matrix multiplication is associative so a chain of matrix multiplications can be done in various orders:



Total computation time  $O(jkl) + O(ijl) = O(jl(i + k))$

$i = k = 1, j = l = 10 \Rightarrow jl(i + k) = 200$

# Non-biological Example – Matrix Chain Multiplication

Finding most efficient order can be done by hierarchical decomposition

$$A_i \cdot A_{i+1} \dots A_j$$

$$= \min_k \left\{ \left( A_i \dots A_k \right) + \left( A_{k+1} \dots A_j \right) \right. \\ \left. + \text{columns in } A_i \times \text{columns in } A_k \times \text{rows in } A_j \right\}$$

# Non-biological Example – Matrix Chain Multiplication

Finding most efficient order can be done by hierarchical decomposition

$$A_i \cdot A_{i+1} \dots A_j$$

$$= \min_k \left\{ \left( A_i \dots A_k \right) + \left( A_{k+1} \dots A_j \right) \right. \\ \left. + \text{columns in } A_i \times \text{columns in } A_k \times \text{rows in } A_j \right\}$$

...or even be formulated as an SCFG optimal parse problem:

$$V_{i,j} \rightarrow \sigma_{i,j} \text{ with probability } \frac{c}{e^{ij}}$$
$$V_{i,j} \rightarrow V_{i,k} V_{k,j} \text{ with probability } \frac{c}{e^{ijk}}$$
$$V_{i,j} \rightarrow \$ \text{ with probability } d_{i,j}$$

where  $c$  is chosen sufficiently small that probabilities sum to at most 1 and  $V_{i,j} \rightarrow \$$  is a dummy production to soak up surplus probability