COUNTING, GENERATING AND SAMPLING TREE ALIGNMENTS

Cedric CHAUVE (Simon Fraser University, Vancouver)
Julien COURTIEL (PIMS/Univ. of British Columbia, Vancouver)
Yann PONTY (CNRS/LIX)

Lille, January 5th 2015
**Motivation:** RNA Comparison

**Question:** how to measure similarity between two RNAs?
MOTIVATION: RNA COMPARISON

Question: how to measure similarity between two RNAs?

First idea: compare nucleic acid sequences.

RNA 1: AUUCGAAUA ... 
RNA 2: ACCAUGAAUA ...
**Motivation: RNA Comparison**

**Question:** how to measure similarity between two RNAs?

First idea: compare nucleic acid sequences. → sequence alignment

RNA 1: AUUCGAAUUA...
RNA 2: ACCAUGAUUUA...

Motivation: RNA Comparison

Question: how to measure similarity between two RNAs?

Second idea: compare secondary structures.

→ notion of tree alignment [Jiang, Wang, Zhang]
FROM SECONDARY STRUCTURES TO TREES
FROM SECONDARY STRUCTURES TO TREES
FROM SECONDARY STRUCTURES TO TREES
FROM SECONDARY STRUCTURES TO TREES
FROM SECONDARY STRUCTURES TO TREES
FROM SECONDARY STRUCTURES TO TREES
FROM SECONDARY STRUCTURES TO TREES

Objective: Align trees coming from RNA 2\textsuperscript{ary} structures
FROM SECONDARY STRUCTURES TO TREES

Objective: Align trees coming from RNA 2ⁿry structures
Trees are plane, rooted, and vertices are labeled by an alphabet $\Sigma$.

Supertree = tree with 3 types of vertices:
- XY (mis)match
- X- insertion
- -Y deletion
Let $A$ be a supertree, 

$$TT_1(A) = \text{tree obtained by changing } \begin{array}{c}
\text{XY} \\
\text{X-} \\
\end{array} \rightarrow \begin{array}{c}
\text{X} \\
\text{X} \\
\end{array}$$

and removing $\begin{array}{c}
\text{-Y} \\
\end{array}$. 

$$TT_2(A) = \text{tree obtained by changing } \begin{array}{c}
\text{XY} \\
\text{-Y} \\
\end{array} \rightarrow \begin{array}{c}
\text{Y} \\
\text{Y} \\
\end{array}$$

and removing $\begin{array}{c}
\text{X-} \\
\end{array}$. 

![Diagram of a tree labeled A with branches labeled C, G, U, and UA.](image-url)
Let $A$ be a supertree,

$\Pi_1(A) =$ tree obtained by changing

$XY \rightarrow \times$

and removing $-Y$.

$\Pi_2(A) =$ tree obtained by changing

$XY \rightarrow Y$

$-Y \rightarrow Y$

and removing $X-$.
Let $A$ be a supertree,

$\Pi_1(A) =$ tree obtained by changing

- $XY$ \rightarrow $X$
- $X\ -$ \rightarrow $X$

and removing $-Y$.

$\Pi_2(A) =$ tree obtained by changing

- $XY$ \rightarrow $Y$
- $-Y$ \rightarrow $Y$

and removing $X\ -$.
Let $A$ be a supertree,

$\mathcal{T}_1(A) =$ tree obtained by changing

- $\text{XY} \rightarrow \text{X}$
- $\text{X-} \rightarrow \text{X}$

and removing $\text{-Y}$.

$\mathcal{T}_2(A) =$ tree obtained by changing

- $\text{XY} \rightarrow \text{Y}$
- $\text{-Y} \rightarrow \text{Y}$

and removing $\text{X-}$.
Let $A$ be a supertree,

$\mathcal{T}_1(A) =$ tree obtained by changing

$\text{XY} \rightarrow \text{XX}$
$\text{X-} \rightarrow \text{XX}$

and removing $\text{-Y}$.

$\mathcal{T}_2(A) =$ tree obtained by changing

$\text{XY} \rightarrow \text{Y}$
$\text{-Y} \rightarrow \text{Y}$

and removing $\text{X-}$.
Let $A$ be a supertree, 

$$\Pi_1(A) = \text{tree obtained by changing } \begin{array}{c} XY \\ X- \end{array} \rightarrow \begin{array}{c} X \\ X \end{array} \text{and removing } -Y. \]

$$\Pi_2(A) = \text{tree obtained by changing } \begin{array}{c} XY \\ -Y \end{array} \rightarrow \begin{array}{c} Y \\ Y \end{array} \text{and removing } X-.
Let $A$ be a supertree,

$$\Pi_1(A) = \text{tree obtained by changing } \begin{array}{l} XY \\ X- \end{array} \rightarrow \begin{array}{l} X \\ X \end{array}$$
and removing $-Y$.

$$\Pi_2(A) = \text{tree obtained by changing } \begin{array}{l} XY \\ -Y \end{array} \rightarrow \begin{array}{l} Y \\ Y \end{array}$$
and removing $X-$. 
Let $A$ be a supertree,

$$\Pi_1(A) = \text{tree obtained by changing } \begin{array}{c} XY \rightarrow \begin{array}{c} X \end{array} \end{array}$$

and removing $-Y$.

$$\Pi_2(A) = \text{tree obtained by changing } \begin{array}{c} XY \rightarrow \begin{array}{c} Y \end{array} \end{array}$$

and removing $X-$. 

\[ \begin{array}{c}
\text{AA} \\
\text{C-} \\
\text{C-} \\
\text{A}
\end{array} \]
Let $A$ be a supertree,

$$\Pi_1(A) = \text{tree obtained by changing } \begin{array}{c} X \ Y \ \X \ - \end{array} \Rightarrow \begin{array}{c} X \ \ \ \ X \ - \end{array} \text{ and removing } -Y.$$ 

$$\Pi_2(A) = \text{tree obtained by changing } \begin{array}{c} X \ Y \ \ -Y \end{array} \Rightarrow \begin{array}{c} Y \ \ \ Y \end{array} \text{ and removing } X-.$$
Let $A$ be a supertree,

\[ \pi_1(A) = \text{tree obtained by changing } \begin{array}{c} XY \\ X- \end{array} \rightarrow \begin{array}{c} X \\ X \end{array} \text{ and removing } -Y. \]

\[ \pi_2(A) = \text{tree obtained by changing } \begin{array}{c} XY \\ -Y \end{array} \rightarrow \begin{array}{c} Y \\ Y \end{array} \text{ and removing } X-. \]
Given two trees $S$ and $T$, a supertree $A$ defines an alignment between $S$ and $T$ if $\Pi_1(A) = S$ and $\Pi_2(A) = T$. 
SUPERTREES INDUCE TREE ALIGNMENTS

Given two trees $S$ and $T$, a supertree $A$ defines an alignment between $S$ and $T$ if $\pi_1(A) = S$ and $\pi_2(A) = T$.

$$\text{cost}(A) = \text{nb of insertions + deletions + mismatches}$$

(can be changed more complicated models)
CONNECTION WITH SEQUENCE ALIGNMENTS

Tree alignments generalize sequence alignments.

AUUCGAUUA... ACCAUGAUA...

alignment:


alignment:

\(AAU- -CU-CC-A-UUGGAAUU...\)
Which alignment between and is the most likely?
Space of Alignments

Which alignment between A
and
is the most likely?

Probability of an alignment A
\( \alpha \propto e^{-\frac{\text{cost}(A)}{K}} \)
(Gibbs-Boltzmann distribution)
Which alignment between and is the most likely?

Probability of an alignment $A$ is $\frac{\exp(-\text{cost}(A))}{\sum_{A'} \exp(-\text{cost}(A'))}$ (Gibbs-Boltzmann distribution)
Why finding one optimal alignment may be inadequate:

- Co-optimal alignments can be very different. (see for instance [Vingron, Argos, 1990])

- Exploring the space of alignments enables the detection of high probability features.
Objective: Sampling alignments under the Gibbs-Boltzmann probability distribution.

probability of an alignment $A$

$\alpha \propto e^{-\text{cost}(A) / K}$

(Gibbs-Boltzmann distribution)

co-optimal alignments
**Objective:** Sampling alignments under the Gibbs-Boltzmann probability distribution.

**Space of Alignments**

- Probability of an alignment $A$: $\alpha \propto e^{-\frac{\text{cost}(A)}{k}}$ (Gibbs-Boltzmann distribution)

Co-optimal alignments
The two supertrees induce the same alignment between the trees.

AMBIGUITY OF ALIGNMENTS
The two supertrees induce the same alignment between the trees.
AMBIGUITY OF ALIGNMENTS

The two supertrees

induce the same alignment between the trees

They are the same!
AMBIGUITY OF ALIGNMENTS

The two supertrees do not induce the same alignment between the trees.
A GRAMMAR FOR ALIGNMENTS

Strategy: Build a context-free grammar that generates every alignment exactly once
A GRAMMAR FOR ALIGNMENTS

Strategy: Build a context-free grammar that generates every alignment exactly once.

An example of grammar that does not work:

\[ G \leftarrow F \]
\[ F \leftarrow \varepsilon \]

[Jiang, Wang, Zhang]
A GRAMMAR FOR ALIGNMENTS

Strategy: Build a context-free grammar that generates every alignment exactly once.

An example of grammar that does not work:

\[ \mathcal{G} \leftarrow \]
\[ F \quad | \quad F \quad | \quad F \]

\[ F \leftarrow \varepsilon \]
\[ F \quad | \quad F \quad | \quad F \quad | \quad F \]

Ex:
\[ e \]
A GRAMMAR FOR ALIGNMENTS

Strategy: Build a context-free grammar that generates every alignment exactly once.

An example of grammar that does not work:

\[ \begin{align*}
E & \leftarrow F \| \overline{F} \| \overline{X} \\
F & \leftarrow \varepsilon \| \overline{F} \| \overline{X} \\
\end{align*} \]

Ex:

\[ e \]
A GRAMMAR FOR ALIGNMENTS

Strategy: Build a context-free grammar that generates every alignment exactly once.

An example of grammar that does not work:

$$E \leftarrow \begin{align*}
F & | \quad X \cdot Y \\
F & | \quad X \cdot - \\
F & | \quad - \cdot Y
\end{align*}$$

$$F \leftarrow \begin{align*}
\varepsilon & | \quad X \cdot Y \\
F & | \quad X \cdot - \\
F & | \quad - \cdot Y
\end{align*}$$

Ex:

$$A \cdot A \downarrow F$$
A GRAMMAR FOR ALIGNMENTS

Strategy: Build a context-free grammar that generates every alignment exactly once.

An example of grammar that does not work:

Ex:
A Grammar for Alignments

Strategy: Build a context-free grammar that generates every alignment exactly once.

An example of grammar that does not work:

\[
G \rightarrow X \cdot Y \mid X \cdot \text{ } F \mid \text{ } F \cdot Y
\]

\[
F \rightarrow \varepsilon \mid X \cdot Y \cdot F \mid X \cdot F \mid F \cdot Y \cdot F \mid F \cdot F
\]

Ex:
A GRAMMAR FOR ALIGNMENTS

Strategy: Build a context-free grammar that generates every alignment exactly once.

An example of grammar that does not work:

\[ G \leftarrow X \]
\[ F \leftarrow \epsilon \]

Ex:

\[ C \]
Strategy: Build a context-free grammar that generates every alignment exactly once.

An example of grammar that does not work:

```
G ← XY | x- | y
F ← ε | XY | x- | y
```

Ex:

```
A

C-   C

F
```

[Jiang, Wang, Zhang]
A GRAMMAR FOR ALIGNMENTS

Strategy: Build a context-free grammar that generates every alignment exactly once

An example of grammar that does not work:

\[ \mathcal{G} \leftarrow \begin{array}{c|c|c}
    \text{XY} & \mathcal{F} & \mathcal{F} \\
    \text{x-} & \mathcal{F} & \mathcal{F} \\
    \text{-y} & \mathcal{F} & \mathcal{F} \\
\end{array} \]

\[ \mathcal{F} \leftarrow \varepsilon \left| \begin{array}{c|c|c|c}
    \text{XY} & \mathcal{F} & \mathcal{F} & \mathcal{F} \\
    \text{x-} & \mathcal{F} & \mathcal{F} & \mathcal{F} \\
    \text{-y} & \mathcal{F} & \mathcal{F} & \mathcal{F} \\
\end{array} \right. \]

Ex:

\[ \begin{array}{c}
    \text{AA} \\
    \text{c-} \quad \text{-c} \quad \mathcal{F} \\
    \mathcal{F} \\
\end{array} \]
A Grammar for Alignments

Strategy: Build a context-free grammar that generates every alignment exactly once.

An example of grammar that does not work:

\[
G \rightarrow A \ |
A \rightarrow XX \ |
A \rightarrow x- \ |
A \rightarrow y \ |
F \rightarrow \varepsilon \ |
F \rightarrow F \ |
F \rightarrow x- \ |
F \rightarrow y \ |
F \rightarrow F
\]

Ex:

\[
\text{Diagram showing an example of an alignment.}
\]
A GRAMMAR FOR ALIGNMENTS

Strategy: Build a context-free grammar that generates every alignment exactly once

An example of grammar that does not work:

Ex:
A GRAMMAR FOR ALIGNMENTS

Strategy: Build a context-free grammar that generates every alignment exactly once.

An example of grammar that does not work:

\[ \mathcal{G} \leftarrow F \]
\[ F \leftarrow \epsilon \]

Ex:

```
A  A
\downarrow \quad \downarrow
C- \quad -G
\downarrow \quad \downarrow
-L \quad C- \quad F
\downarrow \quad \downarrow
F
```
A GRAMMAR FOR ALIGNMENTS

Strategy: Build a context-free grammar that generates every alignment exactly once.

An example of grammar that does not work:

\[ G \leftarrow X \to Y \quad F \quad X \leftarrow - \quad F \quad Y \leftarrow - \quad F \]

\[ F \leftarrow \epsilon \quad X \to Y \quad F \quad X \leftarrow - \quad F \quad Y \leftarrow - \quad F \]

Ex:

```
       A
      / \  
     C-   G
    /     / \
   L     U   C- U A
```
A GRAMMAR FOR ALIGNMENTS

Strategy: Build a context-free grammar that generates every alignment exactly once

An example of grammar that does not work:

\[ G \rightarrow XY \mid x- \mid -y \]

\[ F \rightarrow \epsilon \mid XY \mid F \mid x- \mid F \mid -y \mid F \]

Ex:

\[ AA \quad \epsilon \]

[Jiang, Wang, Zhang]
A GRAMMAR FOR ALIGNMENTS

Strategy: Build a context-free grammar that generates every alignment exactly once

An example of grammar that does not work:

\[ G \leftarrow \begin{align*}
F & \rightarrow XY F \mid x- F \mid -y F \\
F & \rightarrow \epsilon \mid XY F \mid x- F \mid -y F
\end{align*} \]

Ex:

\[ \text{AA} \rightarrow c- d \rightarrow u c- u a \rightarrow F \]
A GRAMMAR FOR ALIGNMENTS

Strategy: Build a context-free grammar that generates every alignment exactly once.

An example of grammar that does not work:

\[ G \leftarrow \begin{array}{c} XY \quad x- \quad -y \\ F \quad F \quad F \end{array} \]

\[ F \leftarrow \begin{array}{c} X \quad Y \\ F \quad F \quad F \end{array} \]

Ex:

\[ \begin{array}{c} AA \\ E- \quad -G \\ \end{array} \]

\[ \begin{array}{c} AA \\ E- \quad -G \\ -U \quad C- \quad UA \end{array} \]
Strategy: Build a context-free grammar that generates every alignment exactly once.

An example of grammar that does not work:

$$G \leftarrow X \rightarrow Y \mid x \rightarrow \epsilon \mid -y \rightarrow F$$

Ex:
A GRAMMAR FOR ALIGNMENTS

Strategy: Build a context-free grammar that generates every alignment exactly once.

An example of grammar that does not work:

\[ G \leftarrow X Y \quad | \quad X - \quad | \quad Y - \]

\[ F \leftarrow \varepsilon \quad | \quad X Y \quad | \quad X - \quad | \quad Y - \]

Ex:

\[ \text{AA} \quad | \quad \text{CC} \quad | \quad \text{UA} \quad | \quad \text{FA} \]

[Jiang, Wang, Zhang]
A GRAMMAR FOR ALIGNMENTS

Strategy: Build a context-free grammar that generates every alignment exactly once.

An example of grammar that does not work:

\[ G \leftarrow \begin{array}{c}
    XY \rightarrow F \\
    x- \rightarrow F \\
    -y \rightarrow F \\
\end{array} \]

\[ F \leftarrow \begin{array}{c}
    \varepsilon \rightarrow F \\
    XY \rightarrow F \\
    x- \rightarrow F \\
    -y \rightarrow F \\
\end{array} \]

Ex:

[Diagram of trees illustrating examples of alignments]
A GRAMMAR FOR ALIGNMENTS

Strategy: Build a context-free grammar that generates every alignment exactly once

An example of grammar that does not work:

\[
G \leftarrow \begin{array}{c}
\text{XY} \\
F
\end{array} \quad \begin{array}{c}
\text{x-} \\
F
\end{array} \quad \begin{array}{c}
\text{-y} \\
F
\end{array}
\]

\[
F \leftarrow \begin{array}{c}
\epsilon \\
\text{XY} \\
F
\end{array} \quad \begin{array}{c}
F
\end{array} \quad \begin{array}{c}
\text{x-} \\
F
\end{array} \quad \begin{array}{c}
\text{-y} \\
F
\end{array}
\]

Ex:
A GRAMMAR FOR ALIGNMENTS

Strategy: Build a context-free grammar that generates every alignment exactly once.

An example of grammar that does not work:

\[ G \leftarrow \begin{array}{c} XY \quad | \quad x- \quad | \quad -y \end{array} \]
\[ F \leftarrow \varepsilon \quad | \quad XY \quad | \quad x- \quad | \quad -y \]

Ex:

ambiguous grammar

[Jiang, Wang, Zhang]
Theorem: The set A generated by the following grammar contains every tree alignment exactly once.
A GRAMMAR FOR ALIGNMENTS

Our (complicated) non-ambiguous grammar:

\[ A \leftarrow \psi \phi \mid C_I \mid C_D \mid \psi \phi \leftarrow \psi \uparrow \mid \psi \uparrow \]

\[ C_I \leftarrow \psi \mid F_\psi \]

\[ C_D \leftarrow \psi \mid F_\psi \]

\[ F_\psi \leftarrow \psi \mid F_\psi \]

\[ \psi \uparrow \leftarrow \psi \mid F_\psi \uparrow F_\psi \]

\[ \psi \uparrow \leftarrow \psi \mid F_\psi \uparrow F_\psi \]

For \( J \in \{ D, D' \} \), \((M, M') \in \{ \phi, \rightarrow, \leftrightarrow \}^2\):

\[ \mathcal{G}_{J, M, M'} \leftarrow \psi \mid C_I \mathcal{G}_{J, M, M'} \mid C_D \mathcal{G}_{J, M, M'} \mid \psi \mathcal{G}_{J, M, M'} \]

only if \( J \neq D \) and \( M \neq \phi \)

no room for \( \mathcal{G}_{J, M, M'} \)
APPLICATION 1: COUNTING.

$a_n = \text{number of tree alignments of size } n$

Generating function:  \[ A(z) = \sum_{n \geq 0} a_n z^n \]
APPLICATION 1: COUNTING.

\[ a_n = \text{number of tree alignments of size } n \]

Generating function: \[ A(z) = \sum_{n \geq 0} a_n z^n \]

The principle on Jiang et al.'s grammar:

\[ \mathcal{G} \leftarrow \]

\[ \mathcal{F} \leftarrow \varepsilon \]
APPLICATION 1: COUNTING.

\[ a_n = \text{number of tree alignments of size } n \]

Generating function:

\[ A(z) = \sum_{n \geq 0} a_n z^n \]

The principle on Jiang et al.'s grammar:

\[ T(z) = F(z) + F(z) + F(z) \]

\[ F(z) = 1 + F(z) \times F(z) + F(z) \times F(z) + F(z) \times F(z) \]
APPLICATION 1: COUNTING.

\[ a_n = \text{number of tree alignments of size } n \]

Generating function: \[ A(z) = \sum_{n \geq 0} a_n z^n \]

The principle on Jiang et al.'s grammar:

\[ T(z) = F(z) + F(z) + F(z) \]

\[ F(z) = 1 + F(z)^2 + F(z)^2 + F(z)^2 \]
APPLICATION 1: COUNTING.

\[ a_n = \text{number of tree alignments of size } n \]

Generating function:
\[ A(z) = \sum_{n \geq 0} a_n z^n \]

The principle on Jiang et al.'s grammar:

\[ C \leftarrow F \uparrow \cdot z^2 \]
\[ T(z) = F(z) + F(z) + F(z) \]

\[ F \leftarrow C \]
\[ F(z) = 1 + F(z)^2 + F(z)^2 + F(z)^2 \]
\( a_m = \text{number of tree alignments of size } n \)

**Generating function:** \( A(z) = \sum_{n \geq 0} a_m z^n \)

The principle on Jiang et al.'s grammar:

\[ \begin{align*}
C & \leftarrow F \quad | \quad X-Y \quad | \quad X-Z \quad | \quad -Y \\
T(z) & = z^2 F(z) + z F(z) + z F(z)
\end{align*} \]

\[ \begin{align*}
F & \leftarrow \varepsilon \quad | \quad X-Y \quad | \quad F \quad | \quad X-Z \quad | \quad -Y \\
F(z) & = 1 + z^2 F(z)^2 + z F(z)^2 + z F(z)^2
\end{align*} \]
APPLICATION 1: COUNTING

\( a_{m,k} \) = number of tree alignments of size \( n \) and \( k \) matches

Generating function: \( A(z, u) = \sum_{n \geq 0} a_{m,k} z^n u^k \)

The principle on Jiang et al.'s grammar:

\[ T(z) = 1 + z^2 F(z) + g_2 x F(z) + g_3 x^2 F(z) \]

\[ F = E \]

The principle on Jiang et al.'s grammar:

\[ F(z) = 1 + z^2 x F(z)^2 + g_2 x F(z)^2 + g_3 x^2 F(z)^2 \]


**Theorem**: The generating function $A(z, u)$ of tree alignments satisfies

$$A(z, u) = \left( z^2 + z - uz^2 + \frac{z^2}{\sqrt{1-4z}} \right) \times B(z, u)$$

where

$$(uz^2C(z) - zC(z)^2 + 2z)B(z, u)^2 + (z^2C^4(z) - 2zC(z)^2 - 1)B(z, u) + C^2(z) = 0$$

and

$$C(z) = \frac{1 - \sqrt{1-4z}}{2z}$$

Catalan generating function
**SOME STATISTICAL PROPERTIES**

**Theorem** There are on average \( C \times 1.5^n \) alignments between two random trees of cumulative size \( n \) where \( C = 0.289 \ldots \)

**Corollary**: A same alignment was repeated \( \sim 0.875 \times 1.412^n \) times on average in the previous ambiguous grammar.
APPLICATION 2 - SAMPLING

Objective: Sampling alignments under the Gibbs-Boltzmann probability distribution.

\[
\alpha \propto e^{-\frac{\text{cost}(A)}{k}}
\]
(Gibbs-Boltzmann distribution)
APPLICATION 2 - SAMPLING

Objective: Sampling alignments under the Gibbs-Boltzmann probability distribution.

Strategy:

→ Filter the grammar to obtain a new grammar that only generates alignments between two fixed trees S and T

→ Use dynamic programming.
The principle on Jiang et al.’s grammar:

\[ G \leftarrow F \quad \text{or} \quad F \leftarrow \varepsilon \]

\[ \begin{align*}
& \quad \text{with different symbols for different trees.}
\end{align*} \]
The principle on Jiang et al.’s grammar:

We fix two trees $S$ and $T$.

Let $F$ be a subforest of $S$ and $G$ a subforest of $T$.

$$J[F, G] = \{ \text{alignments between } F \text{ and } G \}$$

$$G \leftarrow \{ \text{subforests of } S \}$$

$$F \leftarrow \varepsilon \quad \{ \text{subforests of } S \}$$
The principle on Jiang et al.'s grammar:

We fix two trees $S$ and $T$.

Let $F$ be a subforest of $S$ and $G$ a subforest of $T$.

$J[F,G] =$ \# alignments between $F$ and $G$

If $F = \emptyset$ and $G = \emptyset$, then

$J[F,G] \leftarrow \varepsilon$
The principle on Jiang et al.'s grammar:

We fix two trees $S$ and $T$.

Let $F$ be a subforest of $S$ and $G$ a subforest of $T$.

$J[F,G] = \{ \text{alignments between } F \text{ and } G \}$

If $F = \begin{array}{c} F_1 \\ \hline \\ F_2 \end{array}$ and $G = \emptyset$, then

$J[F,G] \leftarrow \begin{array}{c} J[F_2, \emptyset] \\ \hline \\ J[F_1, \emptyset] \end{array}$
The principle on Jiang et al.‘s grammar:

We fix two trees $S$ and $T$.

Let $F$ be a subforest of $S$ and $G$ a subforest of $T$.

$J[F, G] = \{\text{alignments between } F \text{ and } G\}$

If $F = \emptyset$ and $G = G_1 \otimes G_2$, then

$J[F, G] \leftarrow J[\emptyset, G_2] \otimes J[\emptyset, G_1]$
The principle on Jiang et al.'s grammar:

We fix two trees $S$ and $T$.

Let $F$ be a subforest of $S$ and $G$ a subforest of $T$.

$$J[F, G] = \{\text{alignments between } F \text{ and } G\}$$

If $F = F_1 F_2$ and $G = G_1 G_2$, then

$$J[F, G] = J[F_1, G_1] \oplus J[F_2, G_2]$$
GRAMMAR OF ALIGNMENTS BETWEEN TWO FIXED TREES

- If $F = \phi$ and $G = \phi$, then $J[F, G] \leftarrow \varepsilon$
- If $F = \begin{array}{c} F_1 \end{array}$ and $G = \phi$, then
  
  $J[F, G] \leftarrow J[F_2, \phi]$
  
  $J[F_1, \phi]$
- If $F = \phi$ and $G = \begin{array}{c} G_1 \end{array}$, then
  
  $J[F, G] \leftarrow J[\phi, G_1]$
  
  $J[\phi, G_2]$
- If $F = \begin{array}{c} F_1 \end{array}$ and $G = \begin{array}{c} G_1 \end{array}$, then
  
  $J[F, G] \leftarrow J[F_1, G_1]$
  
  $J[F_2, G_1]$
  
  $J[F_1, G_2]$
  
  $J[F_2, G_2]$
Theorem Let $S$ and $T$ be two trees of size $n_1$ and $n_2$. Sampling alignments between $S$ and $T$ under the Gibbs–Boltzmann distribution can be done with worst-case time and space complexities $O(n_1 n_2 (n_1 + n_2)^2)$ and with average-case time and space complexities $O(n_1 n_2)$. 
Theorem Let $S$ and $T$ be two trees of size $n_1$ and $n_2$. Sampling alignments between $S$ and $T$ under the Gibbs-Boltzmann distribution can be done with worst-case time and space complexities $O(n_1 n_2 (n_1 + n_2)^2)$ and with average-case time and space complexities $O(n_1 n_2)$.

Proof inspired by [Herrbach, Denise, Dulucq]
CONCLUSION

→ We are using our grammar and adapted dynamic programming algorithms to revisit the 3D alignments of RNA structures.

→ more general method?
new way to design dynamic programming algorithms?