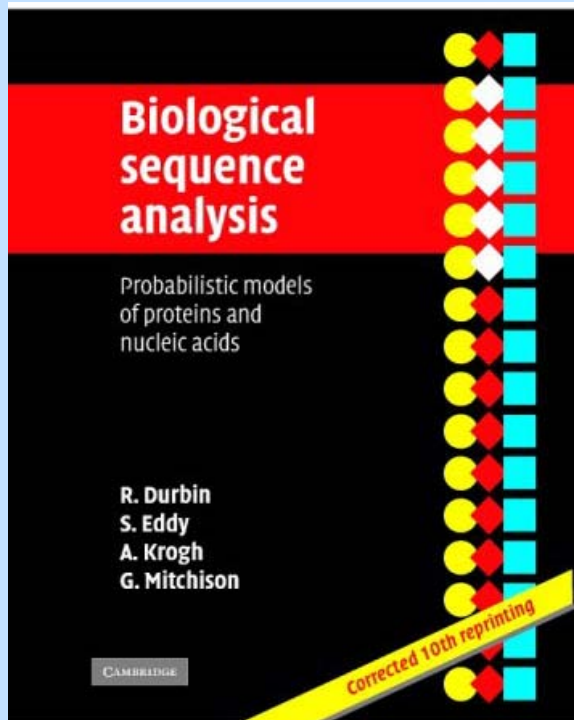


CG, Fall 2011-12, Ron Shamir & Roded Sharan

Stochastic Context Free Grammars



Main source: Durbin et al.,
“Biological Sequence Alignment”
(Cambridge, ‘98)

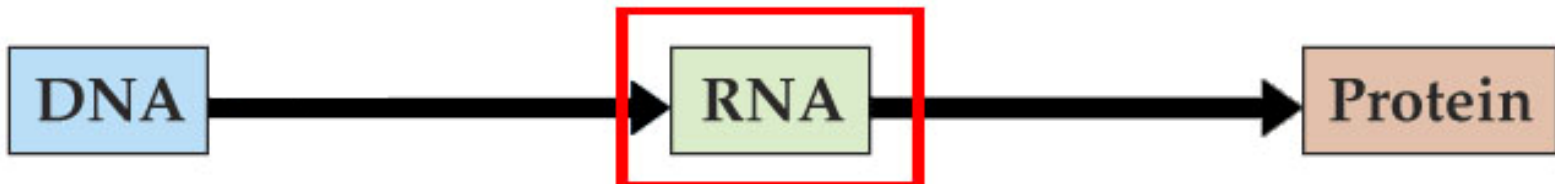
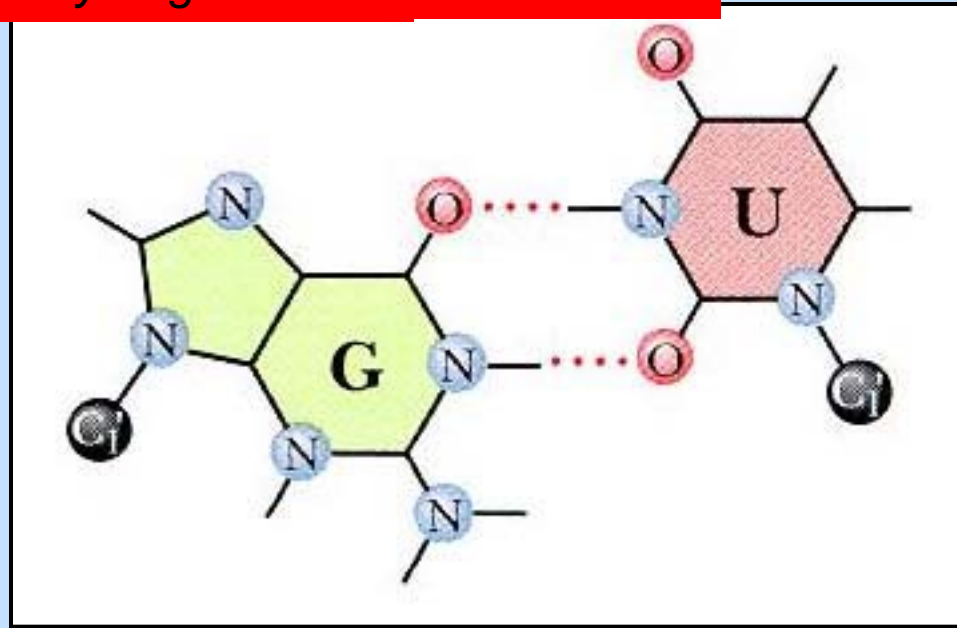
Some slides from B. Majoros



RNA Basics

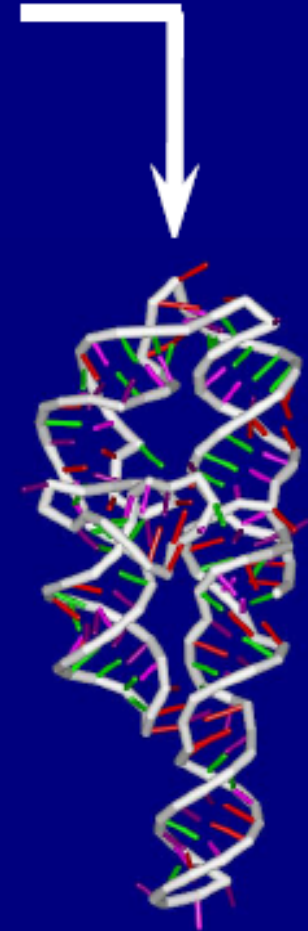
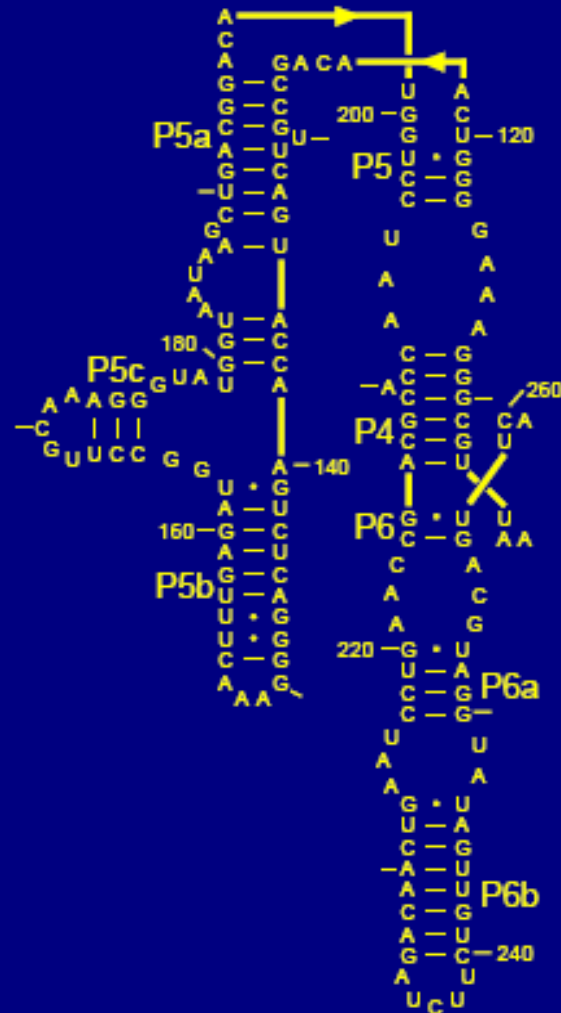
- RNA bases A,C,G,U
- Canonical Base Pairs
 - A-U
 - G-C
 - G-U"wobble" pairing
 - Bases can only pair with **one** other base.

3 Hydrogen Bonds – more stable



RNA Secondary and Tertiary Structure:

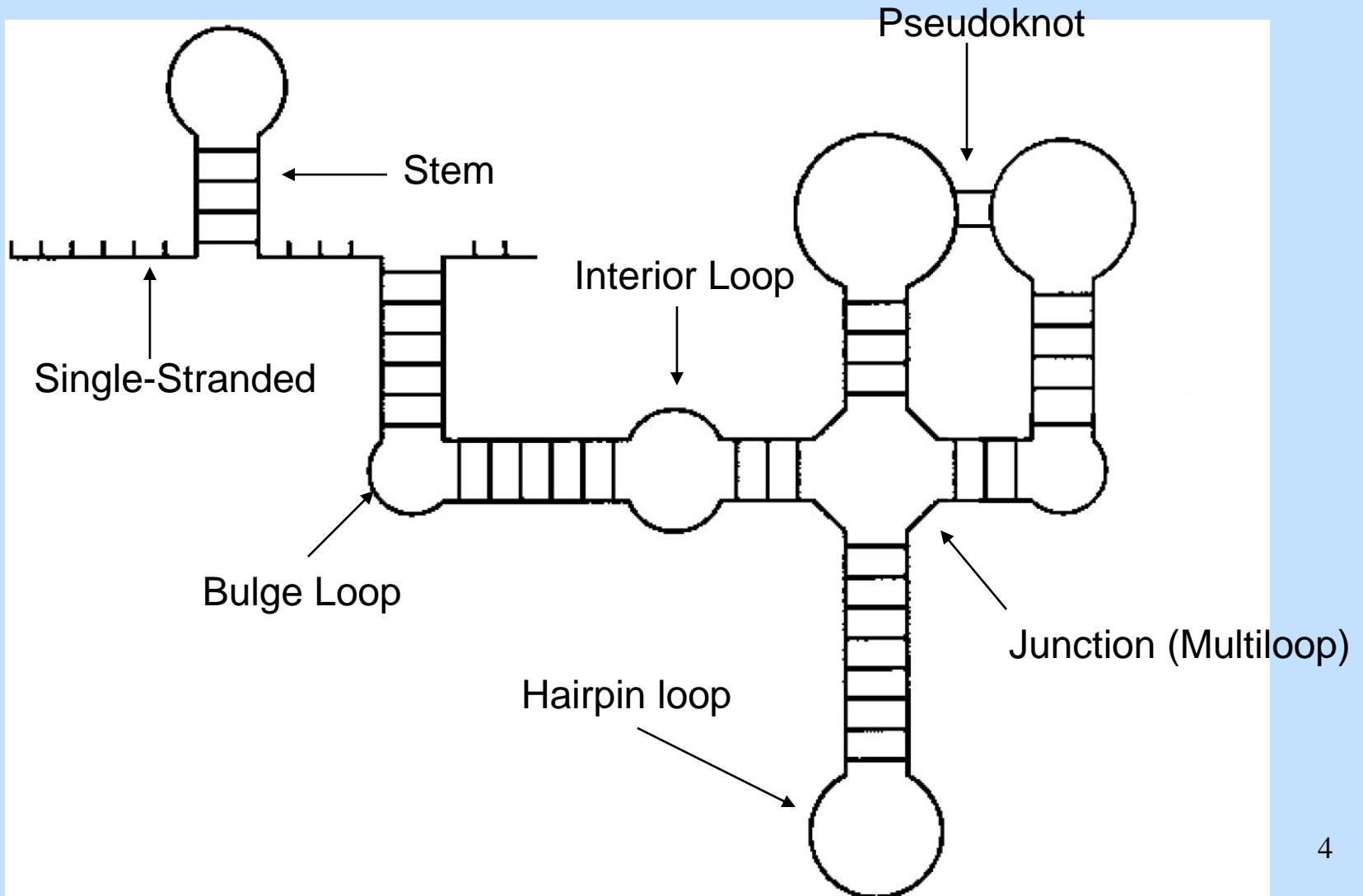
AAUUGCGGGAAAGGGGUCAA
 CAGCCGUUCAGUACCAAGUC
 UCAGGGGAAACUUUGAGAUG
 GCCUUGCAAAGGGUUAUGGUA
 AUAAGCUGACGGACAUGGUC
 CUAACCACGCAGCCAAGUCC
 UAAGUCAACAGAUCUUCUGU
 UGAUAUGGAUGCAGUUCA



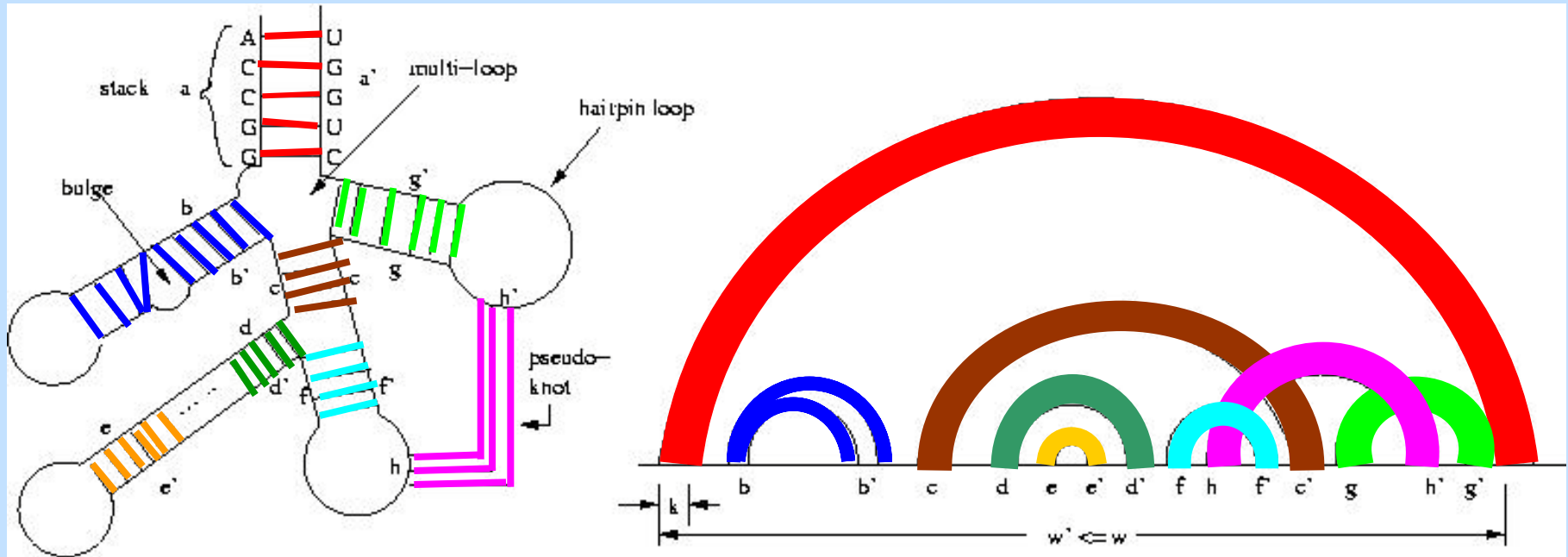
Waring & Davies.
 (1984) *Gene* 28: 277.

Cate, *et al.* (Cech & Doudna).
 (1996) *Science* 273:1678.

RNA Secondary Structure

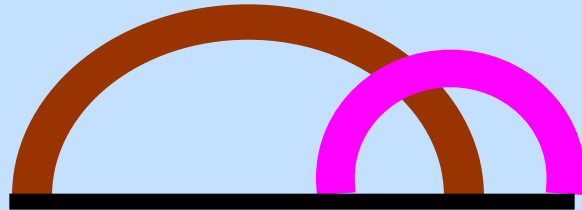


RNA Structure: Details



Pseudoknots

Intersecting base pairs



We shall assume no pseudoknots
in the structure



Grammars



Grammars express languages

Example: **the English language**

$\langle \textit{sentence} \rangle \rightarrow \langle \textit{noun_phrase} \rangle \langle \textit{predicate} \rangle$

$\langle \textit{noun_phrase} \rangle \rightarrow \langle \textit{article} \rangle \langle \textit{noun} \rangle$

$\langle \textit{predicate} \rangle \rightarrow \langle \textit{verb} \rangle$

$\langle \textit{article} \rangle \rightarrow a$

$\langle \textit{article} \rangle \rightarrow \textit{the}$

$\langle \textit{noun} \rangle \rightarrow \textit{boy}$

$\langle \textit{noun} \rangle \rightarrow \textit{dog}$

$\langle \textit{verb} \rangle \rightarrow \textit{runs}$

$\langle \textit{verb} \rangle \rightarrow \textit{walks}$

A derivation of "the boy walks":

$\langle \textit{sentence} \rangle \Rightarrow \langle \textit{noun_phrase} \rangle \langle \textit{predicate} \rangle$
 $\Rightarrow \langle \textit{noun_phrase} \rangle \langle \textit{verb} \rangle$
 $\Rightarrow \langle \textit{article} \rangle \langle \textit{noun} \rangle \langle \textit{verb} \rangle$
 $\Rightarrow \textit{the} \langle \textit{noun} \rangle \langle \textit{verb} \rangle$
 $\Rightarrow \textit{the boy} \langle \textit{verb} \rangle$
 $\Rightarrow \textit{the boy walks}$

Language of the grammar:

$L = \{$ "a boy runs",
"a boy walks",
"the boy runs",
"the boy walks",
"a dog runs",
"a dog walks",
"the dog runs",
"the dog walks" $\}$

Notation

$\langle \textit{noun} \rangle \rightarrow \textit{boy}$

$\langle \textit{noun} \rangle \rightarrow \textit{dog}$

Variable
or
Non-terminal

Production
rule

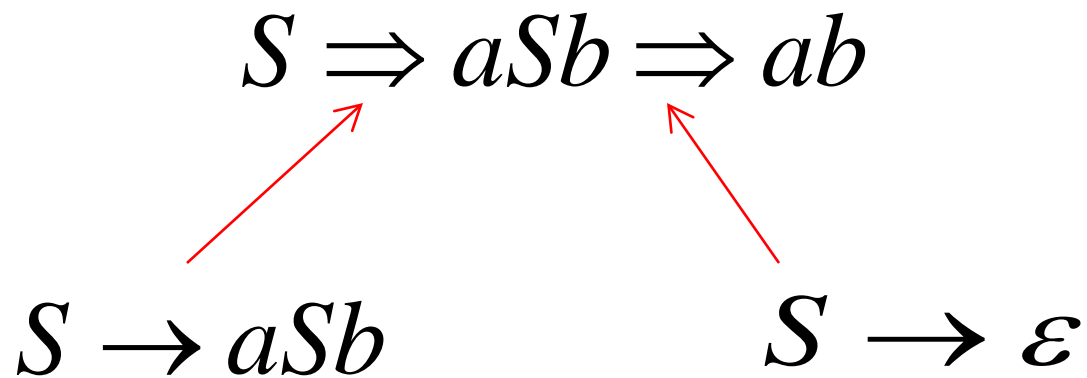
Terminal

Another Example

Grammar: $S \rightarrow aSb$

$S \rightarrow \varepsilon$

Derivation of sentence ab :



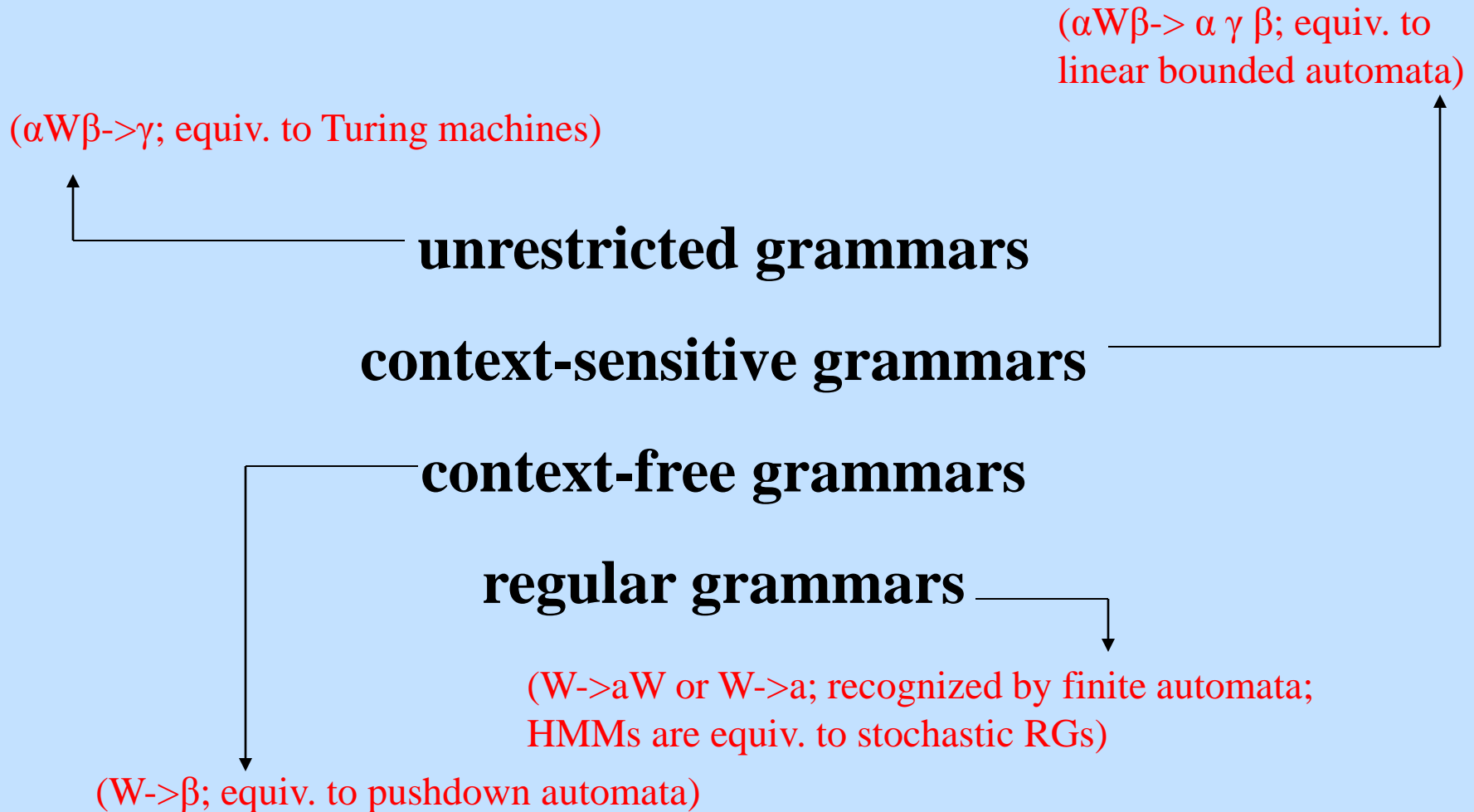
Another derivation:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbbb$$

Language of the grammar

$$L = \{a^n b^n : n \geq 0\}$$

The Chomsky Hierarchy



Context-free Grammars (CFG's)

A *context-free grammar* is a generative model

$$G = (V, \alpha, S, R) \text{ where:}$$

V is a *nonterminal alphabet*, (e.g., $\{A, B, C, D, E, \dots\}$)

α is a *terminal alphabet*, (e.g., $\{a, c, g, t\}$)

$S \in V$ is a special *start symbol*

R is a set of rewriting rules called *productions*.

Productions in R are rules of the form: $X \rightarrow \lambda$

where $X \in V$, $\lambda \in (V \cup \alpha)^*$



Context-freeness

The “*context-freeness*” is imposed by the requirement that the l.h.s of each production rule may contain only a single symbol, and that symbol must be a nonterminal:

$$X \rightarrow \lambda$$

Thus, a CFG cannot specify *context-sensitive* rules such as:

$$wXz \rightarrow w\lambda z$$



Derivations

Suppose a CFG G has generated a *terminal string* $x \in \alpha^*$. A *derivation* $S \Rightarrow^* x$ denotes a possible way for generating x .

A *derivation* (or *parse*) consists of a series of applications of productions from R , beginning with the *start symbol* S and ending with the *terminal string* x :

$$S \Rightarrow s_1 \Rightarrow s_2 \Rightarrow s_3 \Rightarrow \cdots \Rightarrow x$$

where $s_i \in (V \cup \alpha)^*$.

We will concentrate on **leftmost** derivations, where the leftmost nonterminal is always replaced first.



Context-free Versus Regular

The advantage of CFGs over HMMs lies in their ability to model arbitrary runs of matching pairs of elements, such as matching pairs of parentheses:

$$\dots(((((((\dots)))))))))\dots$$

When the number of matching pairs is unbounded, a finite-state model such as an HMM is inadequate to enforce the constraint that all left elements must have a matching right element.

In contrast, in a CFG we can use rules such as $X \rightarrow (X)$. A sample derivation using such a rule is:

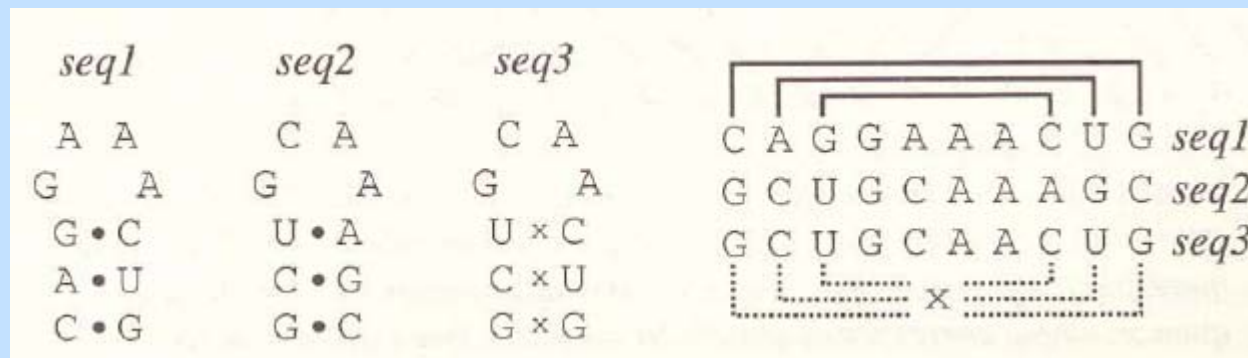
$$X \Rightarrow (X) \Rightarrow ((X)) \Rightarrow (((X))) \Rightarrow ((((X))))$$

An additional rule such as $X \rightarrow \epsilon$ is necessary to terminate the recursion.



A CFG for an RNA stem loop

- Wish a CFG that models RNA stem loops with 3 bps and a GCAA or GAAA loop.



$S \rightarrow aXu \mid cXg \mid gXc \mid uXa$

$X \rightarrow aYu \mid cYg \mid gYc \mid uYa$

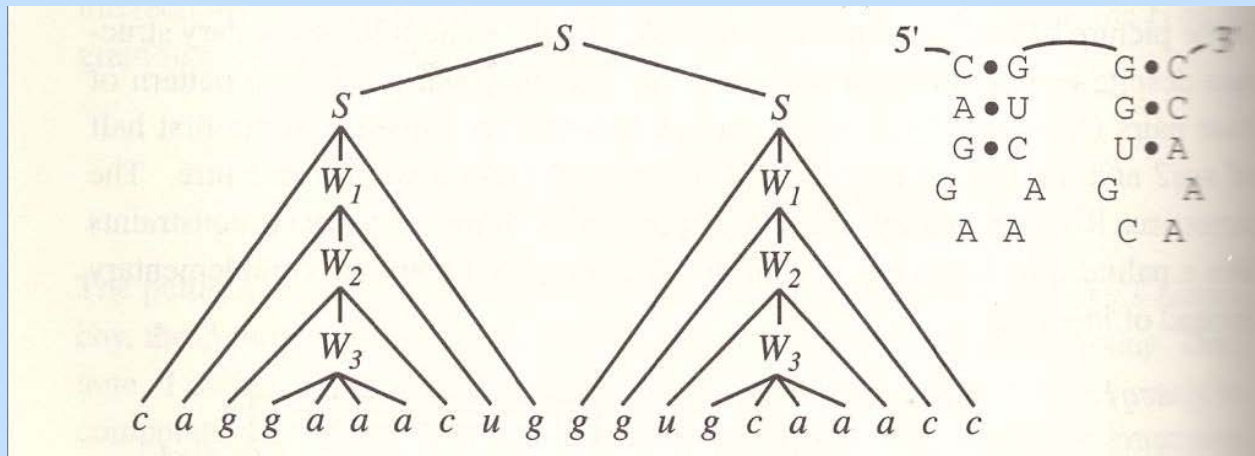
$Y \rightarrow aZu \mid cZg \mid gZc \mid uZa$

$Z \rightarrow gaaa \mid gcaa$



Parse trees

- A representation of a parse of a string by a CFG
- **Root** – start nonterminal S
- **Leaves** – terminal symbols in the given string
- **Internal nodes** - nonterminals
- The children of an internal node are the productions of that nonterminal (left-to-right order)
- **A subtree spans a contiguous sequence segment**



Stochastic CFG

A *stochastic context-free grammar* (*SCFG*) is a CFG plus a probability distribution on productions:

$$G = (V, \alpha, S, R, P_p)$$

where $P_p : R \mapsto [0,1]$, and probabilities are normalized at the level of each nonterminal X :

$$\forall [\sum_{X \in V} \sum_{X \rightarrow \lambda} P_p(X \rightarrow \lambda) = 1]$$

The probability of a derivation $S \Rightarrow^* x$ is the product of the probabilities for all its productions: $\prod_i P(X_i \rightarrow \lambda_i)$

We can sum over all possible (leftmost) derivations of a given string x to get the probability that G will generate x at random: $P(x | G) = \sum_j P(S \Rightarrow_j^* x | G)$.



A Simple Example

As an example, consider $\mathcal{G}=(V_{\mathcal{G}}, \alpha, S, R_{\mathcal{G}}, P_{\mathcal{G}})$, for $V_{\mathcal{G}}=\{S, L, N\}$, $\alpha=\{a, c, g, t\}$, and $R_{\mathcal{G}}$ the set consisting of:

$$S \rightarrow aSt \mid tSa \mid cSg \mid gSc \mid L \quad (P=0.2)$$

$$L \rightarrow NNNN \quad (P=1.0)$$

$$N \rightarrow a \mid c \mid g \mid t \quad (P=0.25)$$

The probability of the sequence $acgtacgtacgt$ is given by:

$$P(acgtacgtacgt) =$$

$$P(S \Rightarrow aSt \Rightarrow acSgt \Rightarrow acgScgt \Rightarrow acgtSacgt \Rightarrow \\ acgtLacgt \Rightarrow acgtNNNNacgt \Rightarrow acgtaNNNacgt \Rightarrow \\ acgtacNNacgt \Rightarrow acgtacgNacgt \Rightarrow acgtacgtacgt) =$$

$$0.2 \times 0.2 \times 0.2 \times 0.2 \times 0.2 \times 1 \times 0.25 \times 0.25 \times 0.25 \times 0.25 = 1.25 \times 10^{-6}$$

because this sequence has only one possible (leftmost) derivation under grammar \mathcal{G} .



Chomsky Normal Form (CNF)

Any CFG which does not derive the empty string (i.e., $\varepsilon \notin L(G)$) can be converted into an equivalent grammar in *Chomsky Normal Form (CNF)*. A CNF grammar is one in which all productions are of the form:

$$X \rightarrow YZ \text{ or } X \rightarrow a$$

for nonterminals X, Y, Z , and terminal a .

Transforming a CFG into CNF can be accomplished by appropriately-ordered application of the following operations (ex.):

- Eliminating *useless symbols* (nonterminals that only derive ε)
- Eliminating *null productions* ($X \rightarrow \varepsilon$)
- Eliminating *unit productions* ($X \rightarrow Y$)
- Factoring long rhs expressions ($A \rightarrow abc$ factored into $A \rightarrow aB, B \rightarrow bC, C \rightarrow c$)
- Factoring terminals ($A \rightarrow cB$ is factored into $A \rightarrow CB, C \rightarrow c$)



CNF - Example

Non-CNF:

$$\begin{aligned} S &\rightarrow a S t \mid t S a \mid c S g \mid g S c \mid L \\ L &\rightarrow N N N N \\ N &\rightarrow a \mid c \mid g \mid t \end{aligned}$$

CNF:

$$\begin{aligned} S &\rightarrow A S_T \mid T S_A \mid C S_G \mid G S_C \mid N L_1 \\ S_A &\rightarrow S A \\ S_T &\rightarrow S T \\ S_C &\rightarrow S C \\ S_G &\rightarrow S G \\ L_1 &\rightarrow N L_2 \\ L_2 &\rightarrow N N \\ N &\rightarrow a \mid c \mid g \mid t \\ A &\rightarrow a \\ C &\rightarrow c \\ G &\rightarrow g \\ T &\rightarrow t \end{aligned}$$

Disadvantages of CNF: (1) more nonterminals & productions, (2) less obvious relation to problem domain

Advantage: easy implementation of parsers



The Parsing Problem

Two questions for a CFG:

- 1) Can a grammar G derive string x ?
- 2) If so, what series of productions would be used during the derivation? (*there may be multiple answers!*)

Additional questions for an SCFG:

- 1) What is the *probability* that G derives string x ?
- 2) What is the *most probable* derivation of x via G ?



The CYK Parsing Algorithm

Given a grammar $G = (V, \alpha, S, R)$ in CNF, we build a DP matrix D s.t. $D_{i,j}$ holds the set of nonterminals that could derive the subsequence $x_i \dots x_j$

$$\text{Initialization: } \forall_{1 \leq i \leq n} D_{i,i} = \{A \mid A \rightarrow x_i \in R\}$$

The remainder of the DP matrix is then computed left-to-right, top-to-bottom:

$$D_{i,j} = \{A \mid A \rightarrow BC \in R, \text{ for some } B \in D_{i,k} \text{ and } C \in D_{k+1,j}, i \leq k < j\}$$

$$\text{Termination: } S \Rightarrow^* x \text{ iff } S \in D_{1,n}.$$

(Cocke and Schwartz, 1970; Younger, 1967; Kasami, 1965)



The CYK Parsing Algorithm

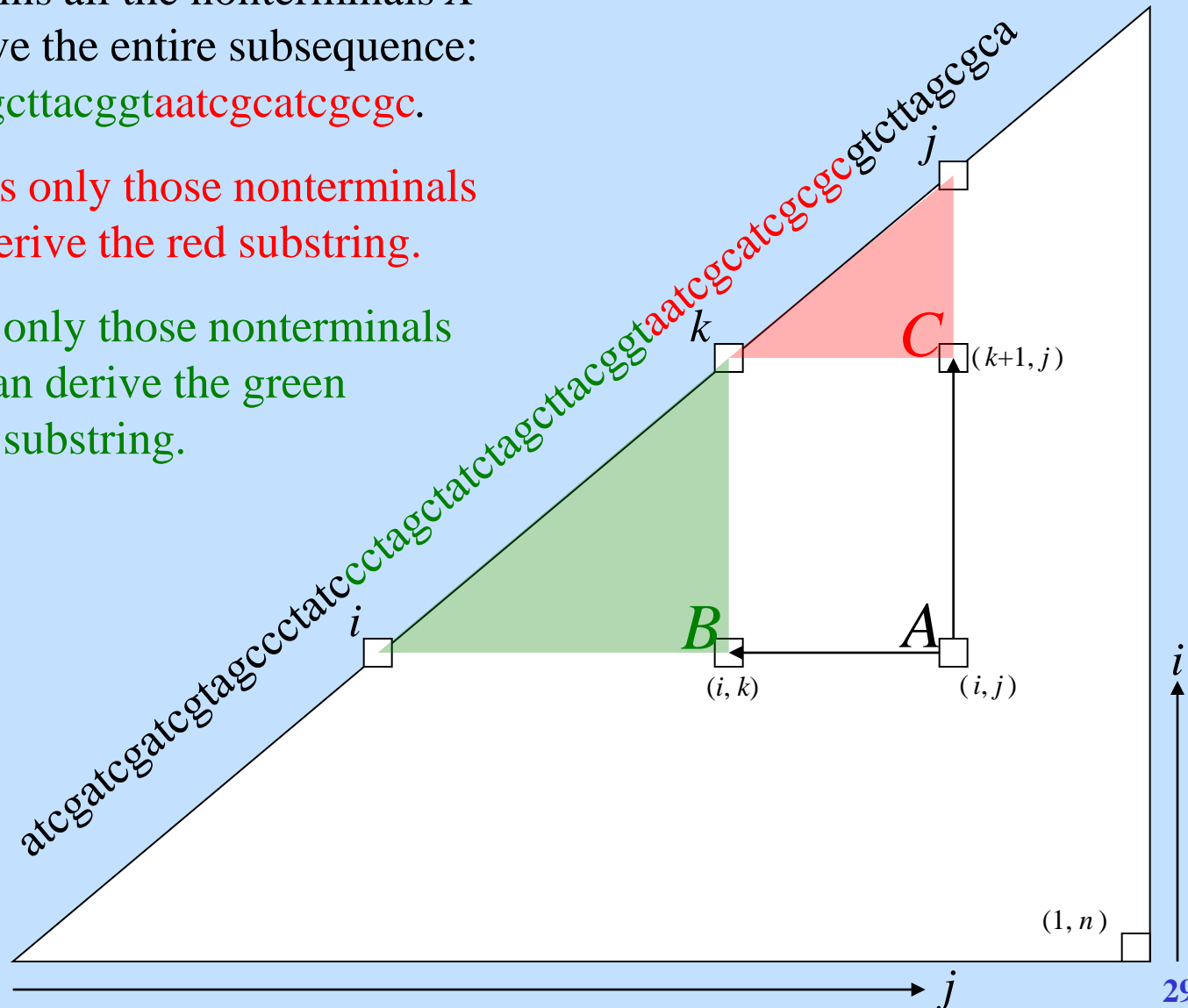
Cell (i, j) contains all the nonterminals X which can derive the entire subsequence:

actagctatctagcttacggtaatcgcacgcgc.

$(k+1, j)$ contains only those nonterminals which can derive the red substring.

(i, k) contains only those nonterminals which can derive the green substring.

$A \rightarrow BC$



CYK for SCFG

$D(i,j,v)$ – (log) probability of optimal parse tree with root v of $x_i \dots x_j$

Initialization: $\forall_{1 \leq i \leq n, v} D(i,i,v) = \log P(v \rightarrow x_i)$

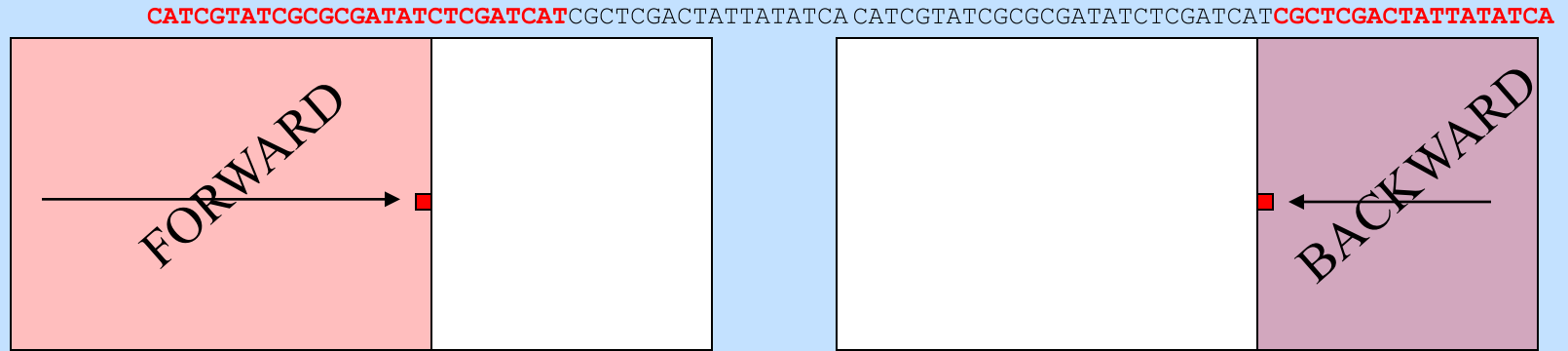
Iteration: $D(i,j,v) = \max_{\{y,z,k\}} [D(i,k,y) + D(k+1,j,z) + \log P(v \rightarrow yz)]$

Termination: $\log P(x, \Pi^*) = D(1,L,S)$

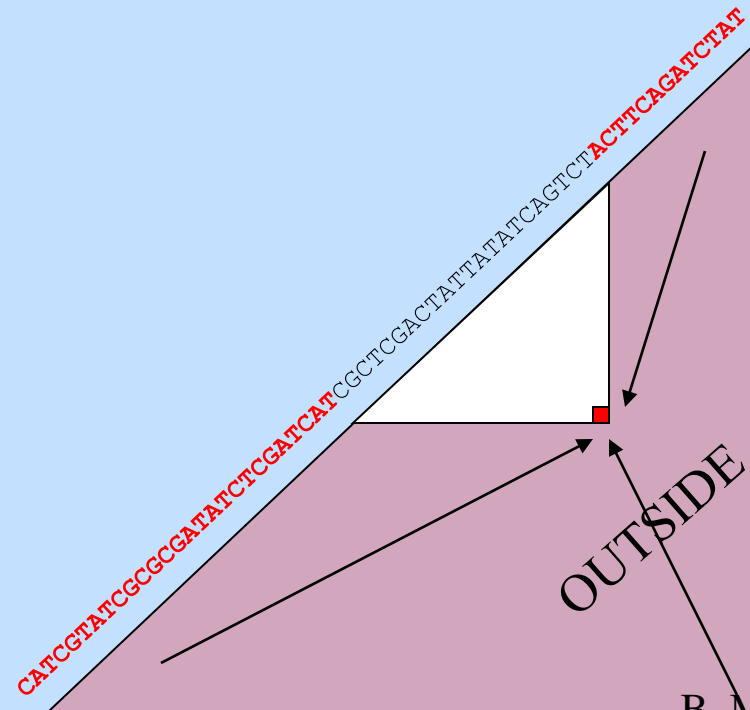
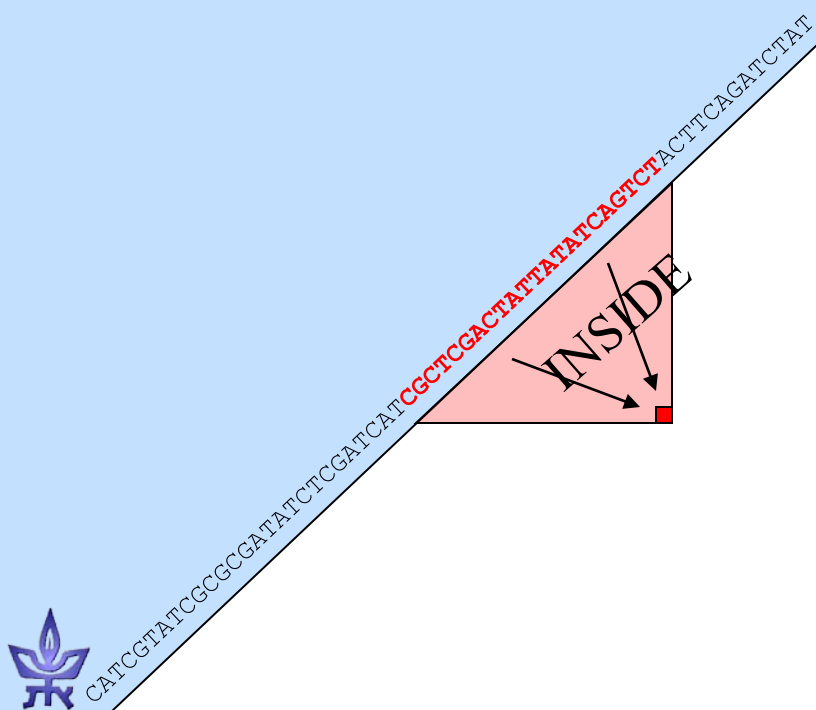
Complexity: $O(L^3M^3)$ time; $O(L^2M)$ memory



Recall: Forward-Backward



Inside-Outside is the equivalent for trees



The Inside Algorithm

The “equivalent” of the forward alg. for evaluating the probability of a parse subtree rooted at nonterminal X for the subsequence $x_i \dots x_j$

$$\alpha(i, j, X) = P(X \Rightarrow^* x_i \dots x_j) = P(x_i \dots x_j / X_{ij}, G)$$

```
for i=1 up to L do
  foreach nonterminal X do
     $\alpha(i, i, X) = P(X \rightarrow x_i)$ ;
for i=L-1 down to 1 do
  for j=i+1 up to L do
    foreach nonterminal X do
       $\alpha(i, j, X) = \sum_{Y, Z} \sum_{k=i \dots j-1} P(X \rightarrow YZ) \alpha(i, k, Y) \alpha(k+1, j, Z)$ ;
```

The probability $P(x|G)$ of the full input sequence x of length L can then be found in the final cell of the matrix: $\alpha(1, L, S)$.

Complexity: $time = O(L^3 M^3)$ $memory = O(L^2 M)$



The Outside Algorithm

Calculates the prob. $\beta(i, j, X)$ of a complete parse tree rooted at S for the complete sequence x , excluding the parse subtree for subsequence $x_i..x_j$ rooted at X .

$$\beta(i, j, X) = P(S \Rightarrow^* x_1..x_{i-1} X x_{j+1}..x_L) = P(x_1..x_{i-1}; X_{ij}; x_{j+1}..x_L | G)$$

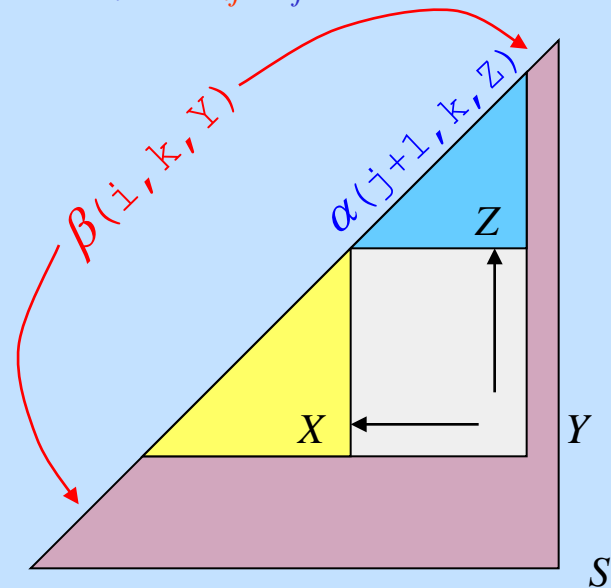
```

 $\beta(1, L, S) = 1;$ 
foreach  $X \neq S$  do  $\beta(1, L, X) = 0;$ 
for  $i = 1$  up to  $L$  do
  for  $j = L$  down to  $i$  do
    foreach nonterminal  $X$  do

```

$$\beta(i, j, X) = \sum_{Y, Z} \sum_{k=j+1..L} P(Y \rightarrow XZ) \alpha(j+1, k, Z) \beta(i, k, Y) + \sum_{Y, Z} \sum_{k=1..i-1} P(Y \rightarrow ZX) \alpha(k, i-1, Z) \beta(k, j, Y);$$

Termination (for any i): $P(x | G) = \sum_X P(X \rightarrow x_i) \beta(i, i, X)$



Training an SCFG

Two common methods for training an SCFG:

- 1) If parses are known for the training sequences, we can simply count the number of times each production occurs in the training parses and normalize these counts into probabilities (analogous to HMMs).
- 2) If parses are NOT known for the training sequences, we can use an EM algorithm similar to the *Forward-Backward* (“Baum-Welch”) algorithm for HMMs. The EM algorithm for SCFGs is called *Inside-Outside*.



Inside-Outside Parameter Estimation

EM-update equations (ex.):

$$P(X_{i,j} | x, G) = \frac{1}{P(x | G)} \beta(i, j, X) \alpha(i, j, X)$$

$$P(X_{i,j} \rightarrow YZ | x, G) = \frac{1}{P(x | G)} \sum_{k=i..j-1} \beta(i, j, X) P(X \rightarrow YZ) \alpha(i, k, Y) \alpha(k+1, j, Z)$$

$$P_{new}(X \rightarrow YZ) = \frac{E(X \rightarrow YZ)}{E(X)} = \frac{\sum_{i=1}^{L-1} \sum_{j=i+1}^L \sum_{k=i}^{j-1} \beta(i, j, X) P(X \rightarrow YZ) \alpha(i, k, Y) \alpha(k+1, j, Z)}{\sum_{i=1}^L \sum_{j=i}^L \beta(i, j, X) \alpha(i, j, X)}$$

$$P_{new}(X \rightarrow a) = \frac{\sum_{i:x_i=a} \beta(i, i, X) P(X \rightarrow a)}{\sum_{i=1}^L \sum_{j=i}^L \beta(i, j, X) \alpha(i, j, X)}$$



Covariance models (Eddy & Durbin 1994)

- A general modeling scheme for RNA families
- Based on “profile” SCFG:
 - Model base pairs and single-stranded positions in an RNA secondary consensus
 - Allow for insertions and deletions w.r.t. the consensus



The basic model

State type	Description	Production	Emission	Transition
P	(pair emitting)	$P \rightarrow aYb$	$e_v(a,b)$	$t_v(Y)$
L	(left emitting)	$L \rightarrow aY$	$e_v(a)$	$t_v(Y)$
R	(right emitting)	$R \rightarrow Ya$	$e_v(a)$	$t_v(Y)$
B	(bifurcation)	$B \rightarrow SS$		
S	(start)	$S \rightarrow Y$		$t_v(Y)$
E	(end)	$E \rightarrow \varepsilon$		



A Toy RNA Family

input multiple alignment:

```
[structure] . x x > > > x x x x < x < < x > > x > . x x x . < < < .
human      . AAGACUUCGGAUCUGGCG . ACA . CCC .
mouse     a UACACUUCGGAUG - CACC . AAA . GUG a
orc       . AGGUCUUC - GCACGGGCAgCCA cUUC .
           1         5         10        15        20        25        28
```

example structure:

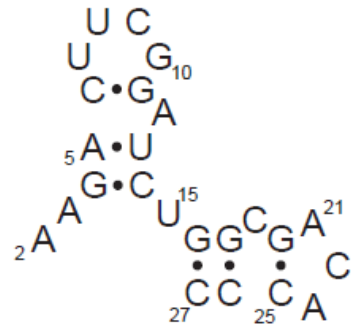


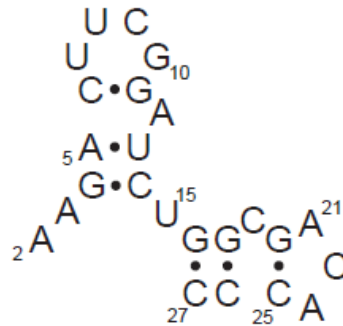
Figure 1

An example RNA sequence family. Top: a toy multiple alignment of three sequences, with 28 total columns, 24 of which will be modeled as consensus positions. The [structure] line annotates the consensus secondary structure: > and < symbols mark base pairs, x's mark consensus single stranded positions, and .'s mark "insert" columns that will not be considered part of the consensus model. Bottom: the secondary structure of the "human" sequence.



An SCFG of the consensus

example structure:



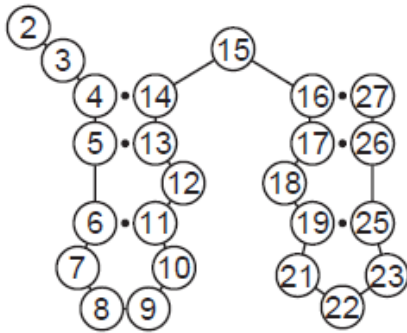
	Stem 1	Stem 2
$S_1 \rightarrow L_2 \dots$	$S_5 \rightarrow P_6$	$S_{15} \rightarrow L_{16}$
$L_2 \rightarrow aL_3 \dots$	$P_6 \rightarrow gP_7c \dots$	$L_{16} \rightarrow uP_{17} \dots$
$L_3 \rightarrow aB_4 \dots$	$P_7 \rightarrow aR_8u \dots$	$P_{17} \rightarrow gP_{18}c \dots$
$B_4 \rightarrow S_5S_{15}$	$R_8 \rightarrow P_9a \dots$	$P_{18} \rightarrow gL_{19}c \dots$
	$P_9 \rightarrow cL_{10}g \dots$	$L_{19} \rightarrow cP_{20} \dots$
	$L_{10} \rightarrow uL_{11} \dots$	$P_{20} \rightarrow gL_{21}c \dots$
	$L_{11} \rightarrow uL_{12} \dots$	$L_{21} \rightarrow aL_{22} \dots$
	$L_{12} \rightarrow cL_{13} \dots$	$L_{22} \rightarrow cL_{23} \dots$
	$L_{13} \rightarrow gE_{14} \dots$	$L_{23} \rightarrow aE_{24} \dots$
	$E_{14} \rightarrow \epsilon$	$E_{24} \rightarrow \epsilon$

- The positional nonterminals are connected by a tree!

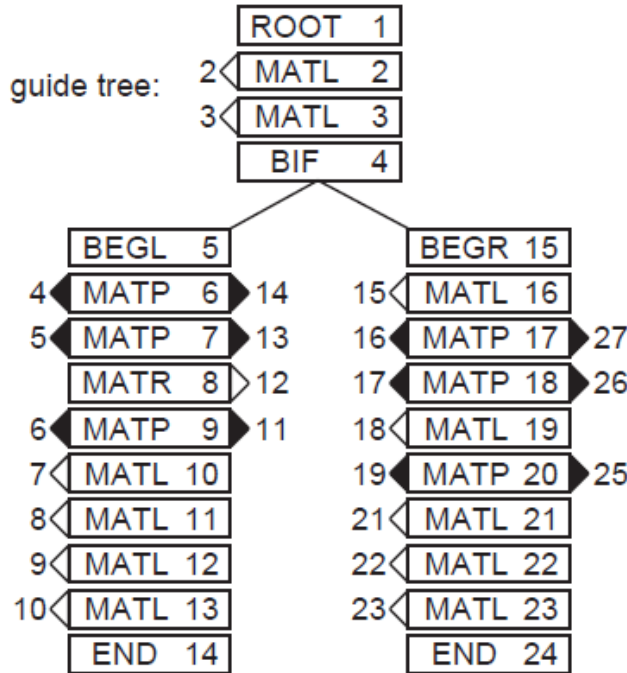


From consensus structure to a guide tree model

consensus structure:



guide tree:



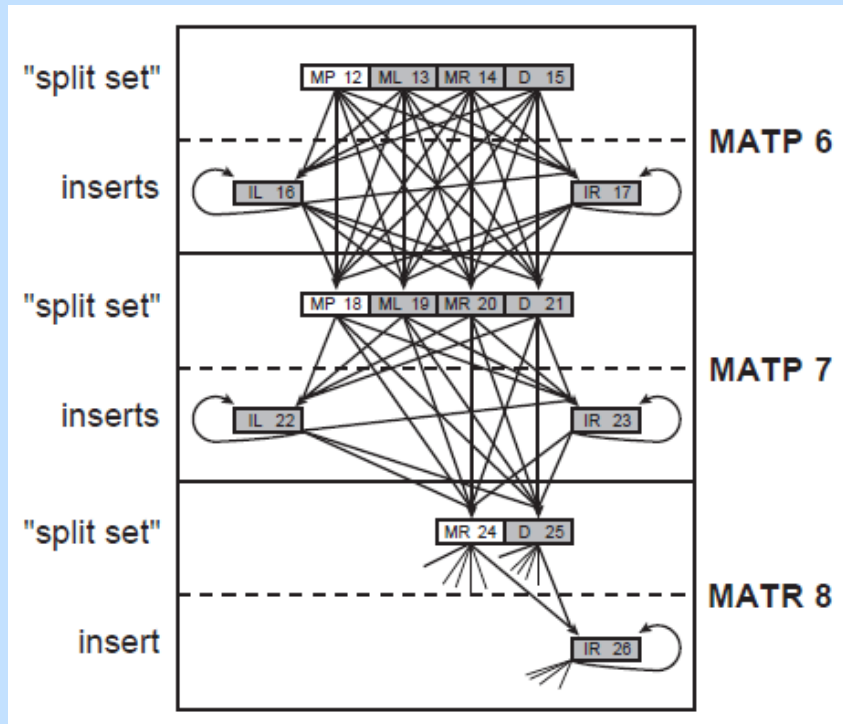
Node	Description
MATP	(pair)
MATL	(single strand, left)
MATR	(single strand, right)
BIF	(bifurcation)
ROOT	(root)
BEGL	(begin, left)
BEGR	(begin, right)
END	(end)

Prefer MATL over MATR

- Non-terminal=state.
- State transition prob. = 1
- Emission probs per state: MATP (16), MATL/R (4)



From guide tree to CM



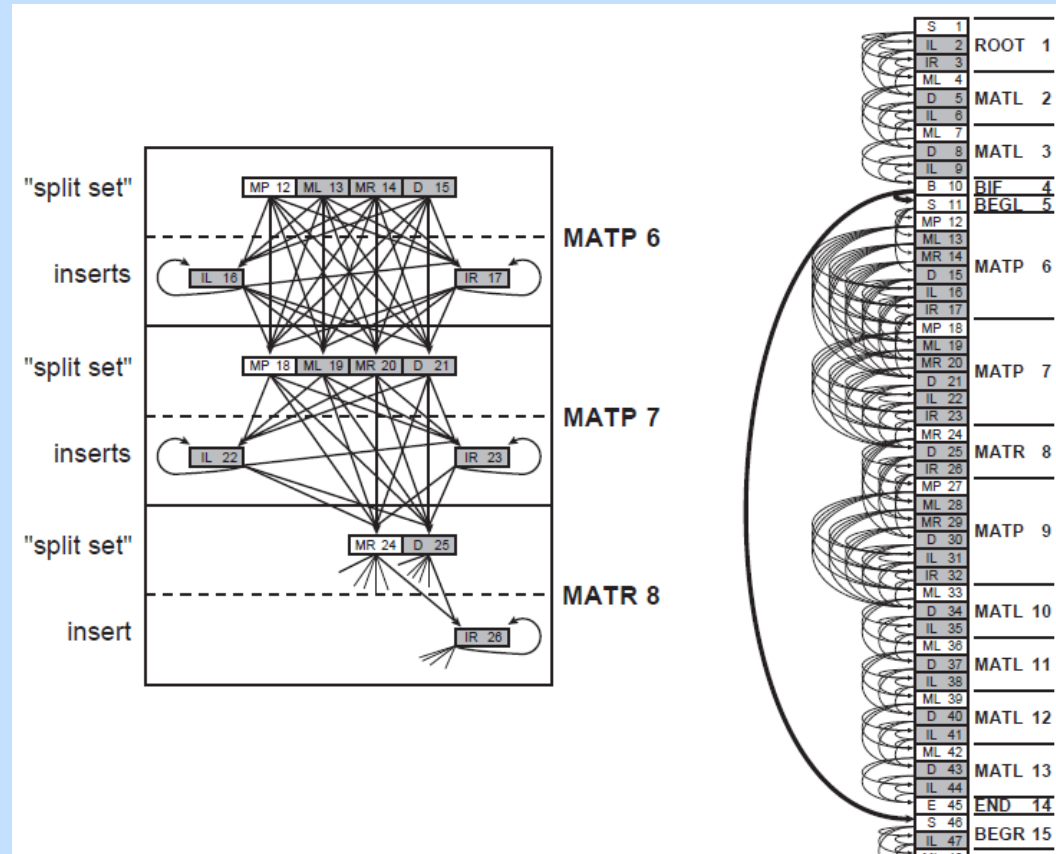
Node	States
MATP	[MP ML MR D] IL IR
MATL	[ML D] IL
MATR	[MR D] IR
BIF	[B]
ROOT	[S] IL IR
B EGL	[S]
B EG R	[S] IL
END	[E]

- Each tree node is expanded into several states:
 - Split set – the main consensus state; one of which must be visited
 - Insert set – visited 0 or more times.



The final CM

- A split state can transition to an insert state in the same node or a split state in the next node
- An IL state can transition to itself, an IR in the same node or a split state in the next
- An IR can transition to itself or to the “next” split state



What if no consensus structure is known ?

1. An initial model is built from a (possibly random or sequence-based) alignment of the training sequences
 - a) Consider columns with >50% gaps as indels.
 - b) The consensus structure is chosen to maximize the mutual information between consensus alignment columns:

$$S_{i,j} = \max \{ S_{i+1,j}, S_{i,j-1}, S_{i+1,j-1} + M_{ij}, \max_{i < k < j} [S_{i,k} + S_{k+1,j}] \}$$

2. The model's parameters are optimized by EM: specialized CM inside-outside & update algorithms are described in pp. 286-8.
3. A new multiple alignment is derived and fed to (1) till convergence.



Performance Evaluation I

- Two training modes: (A) start from known alignment; (U) start from unaligned sequences
- Bit log-odds score averaged over test set
- Alignment accuracy – % truly aligned symbol pairs that are also aligned in the inferred alignment

Mode	#Iterations	Bit score	Accuracy
A100	3	57.3	94%
U100	23	56.7	90%

30 for
HMM

30% for
degapped
alignment



Multiple sequence alignment

- Example of yeast tRNAs whose 3D structure is known from crystallography.

Trusted:

```
DF6280 GCGGADUUASDCUCAGUU GGG AGAGCGCCAGACUGAAG AUCUGGAG GUCCUQUUUCGAUCCACAGAAUUCACCA
DF6280G BCDRAUUUAQCUCAGUU GGG AGAGCGCCAGACUGAAGAAAUAUCUUCGGUCAAGUUAUCUGGAG GUCCUQUUUCGAUCCACAGAAUUCGCA
DD6280 UCCUGGUAUAGUUUAU GGUCAGAAUGGCGCUCUUGUC CUGGCCAG AUCUGGAGUUCAAUUCUCCGUCUUCGAGCCA
DX1661 CCGCGGUGGAGACAGCCUGGU AGCUCGUCGCGCUCAUA ACCCGAAG GUCCUQUUUCGAUCCACAGAAUUCACCA
DS6280 UUCAACUUGSCCAGU GGUUAAGGCGAAGAUUAGAA AUCUUU GGGCUUUGCCCG CUCAGUUCGAGUCCUUCAGUUUCGCCA
```

U100:

```
DF6280 GCGGADUUAGCCACAG UUGGGAGGCGCCAGACU GA AG AUCUGGA GUCCUQUUUCGAUCCACAGAAUUCACCA
DF6280G GCGGADUUAGCCACAG UUGGGAGGCGCCAGACU GaaagaaauacuUCgguCAaguuAUCUGGA GUCCUQUUUCGAUCCACAGAAUUCGCA
DD6280 UCCUGGUAUAGUUUA UGGUCAGAAUGGCGCUCU GU CG CUGGCCA GAU CCGGUGUUCAAUUCUCCGUCUUCGAGCCA
DX1661 CCGCGGUGGAGACAGC CUGGUAGCUCGCGCU CA UA ACCCGAA GUCCUQUUUCGAUCCACAGAAUUCACCA
DS6280 UUCAACUUGSCCAG UGGUUAAGGCGAAGAUU AG AA AUCUUU GgggcuuugcccG CUCAGUUCGAGUCCUUCAGUUUCGCCA
```

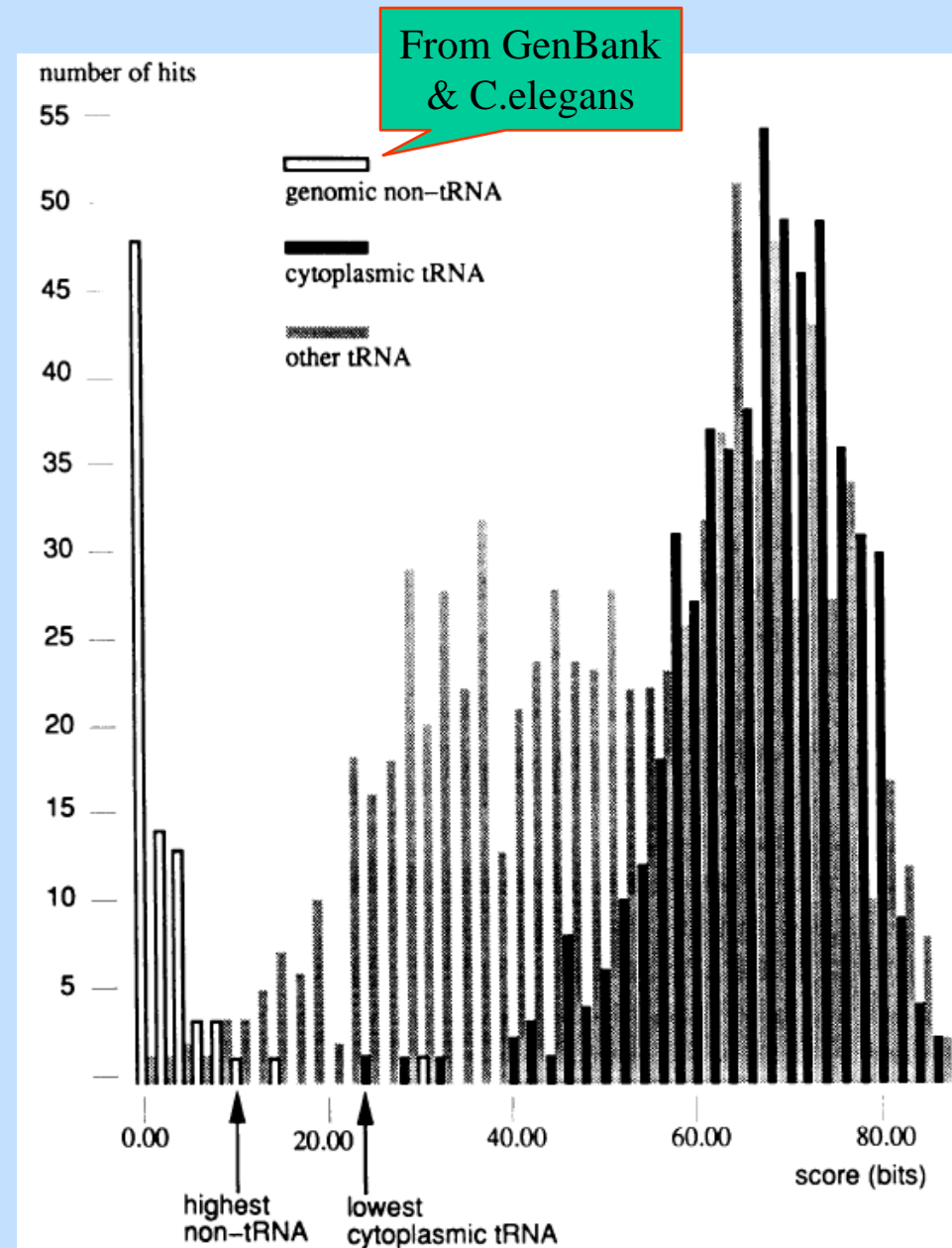
ClustalV:

```
DF6280 GCGGADUUASDCUCAGUUGGGAGAGCGCCAGACUGAAGA UCUGGAGGUCCUQUUUCGAUCCACAGAAUUCACCA
DF6280G BCDRAUUUAQCUCAGUUGGGAGAGCGCCAGACUGAAGAAAUAUCUUCGGUCAAGUUAUCUGGAGGUCCUQUUUCGAUCCACAGAAUUCGCA
DD6280 UCCUGGUAUAGUUUAU G GUCAGAAUGGCGCUCUUG UUG UCGCGGCC AGAUUCG UCUUCAAUUCUCCGUCUUCGAGCCA
DX1661 CCGCGGUGGAGACAGC CUGGUAGCUCGCGCU CUCA UAACCGAA AGGUCUQUUUCGAUCCACAGAAUUCACCA
DS6280 UUCAACUUGSCCAGUGGUUAAGGCGAAGAUU AGAAUCUUUGGGC UUUGCCCG CUCAGUUCGAGUCCUUCAGUUUCGCCA
```



Performance Evaluation II

- Tested A1415 in searching 6Mb sequence from GenBank structural RNA db and *C. elegans* genome.
- Perfect separation for a wide threshold range of the more conserved c-tRNAs
- In *C. elegans* all 14 tRNAs were detected (score > 31), no false positives
- In, 26/522 (5%) annotated tRNAs were missed; 22/26 lack the D-stem loop



Rfam 9.1 (January 2009, 1372 families)

The Rfam database is a collection of RNA families, each represented by **multiple sequence alignments**, **consensus secondary structures** and **covariance models (CMs)**. [More...](#)

QUICK LINKS

SEQUENCE SEARCH

VIEW AN RFAM FAMILY

KEYWORD SEARCH

TAXONOMY SEARCH

JUMP TO

YOU CAN FIND DATA IN RFAM IN VARIOUS WAYS...

Analyze your RNA sequence for Rfam matches

View Rfam family annotation and alignments

Query Rfam by keywords

Fetch families or sequences by NCBI taxonomy

Enter any type of accession or ID to jump to the page for a Rfam family, sequence or genome

Or view the [help](#) pages for more information

