Kapitel 1
Genvorhersage

Vorlesung *Algorithmen der Bioinformatik II* vom 27 und 29. April 2010

Dr. Mario Stanke
Institut für Mikrobiologie und Genetik
Universität Göttingen

These Slides Available At:

http://gobics.de/mario/AbiII

The study aims of this week.

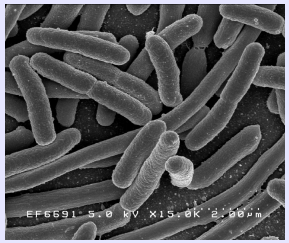
1. understand the problem setting of gene finding
2. learn about algorithmic solutions: exon chaining, GHMMs
3. learn about *pair HMMs*
   (used both for gene finding and alignments)

Overview

Prokaryotes, Eukaryotes

Prokaryotes

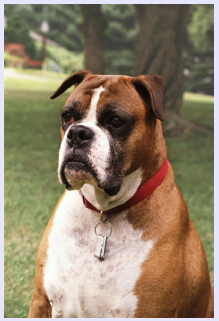
Prokaryotes are the set of species that lack a cell nucleus.
{prokaryotes} = {bacteria} ∪ {archea}



Eukaryotes

Eukaryote are the set of species whose cells have a nucleus.
May be unicellular (e.g. some algae) or multicellular (plants and animals).

Copyright by Jim Pisarowicz



Copyright by Broad Institute

Prokaryotes, Eukaryotes

- the structure of prokaryotic genes is less complex than those of eukaryotes.

Prokaryotes, Eukaryotes

- the structure of prokaryotic genes is less complex than those of eukaryotes.
- prokaryotic gene finding is
  - easier,
  - algorithmically less interesting
  - and can be considered a special case (missing introns).

Prokaryotes, Eukaryotes

- the structure of prokaryotic genes is less complex than those of eukaryotes.
- prokaryotic gene finding is
  - easier,
  - algorithmically less interesting
  - and can be considered a special case (missing introns).
- We will therefore restrict lecture to eukaryotes

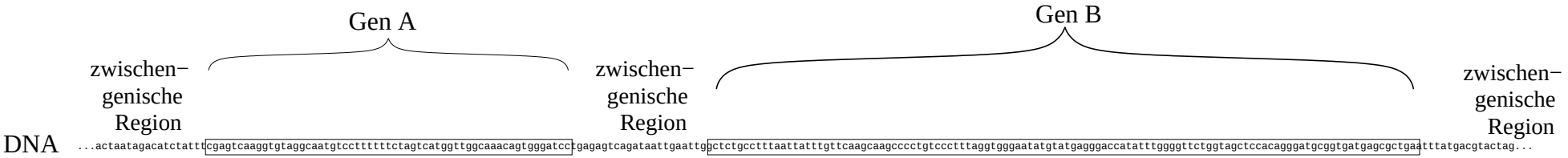
Structure of a eukaryotic gene

DNA ...actaatagacatctatttcgagtcaaggtgtaggcaatgtcctttttctagtcatggttggcaaacagtgggatcctgagagtcagataattgaattggctctgcctttaattatttgttcaagcaagcccctgtccctttaggtgggaatatgtatgagggaccatatttggggttctggtagctccacagggatgcggtgatgagcgctgaatttatgacgtactag...

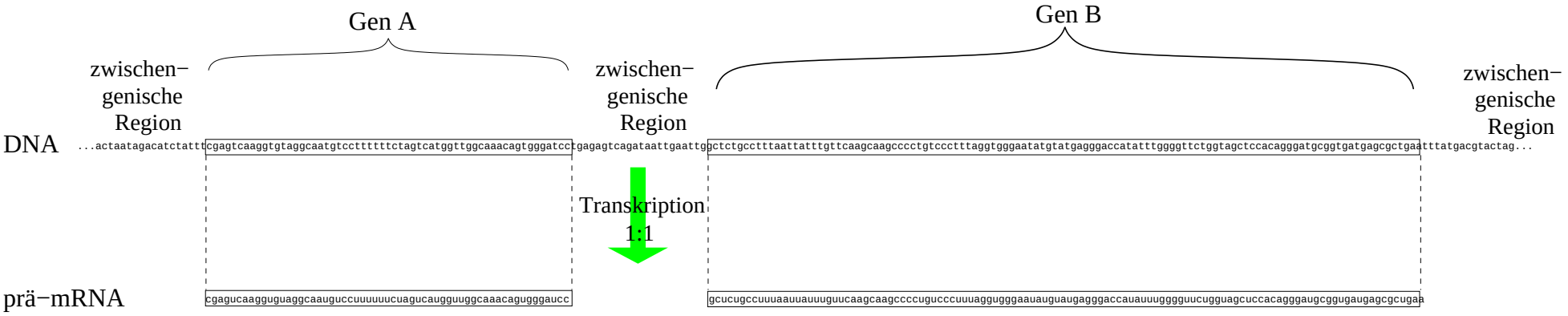
Structure of a eukaryotic gene

DNA ...actaatagacatctatttcgagtcaaggtgtaggcaatgtccttttttctagtcatggttggcaaacagtg gaattggc aattatttgttcaagcaagcccctgtccctttaggtgggaatatgtatgagggaccatatttggggttctggtagctccacagggatgcggtgatgagcgctgaatttatgacgtactag...

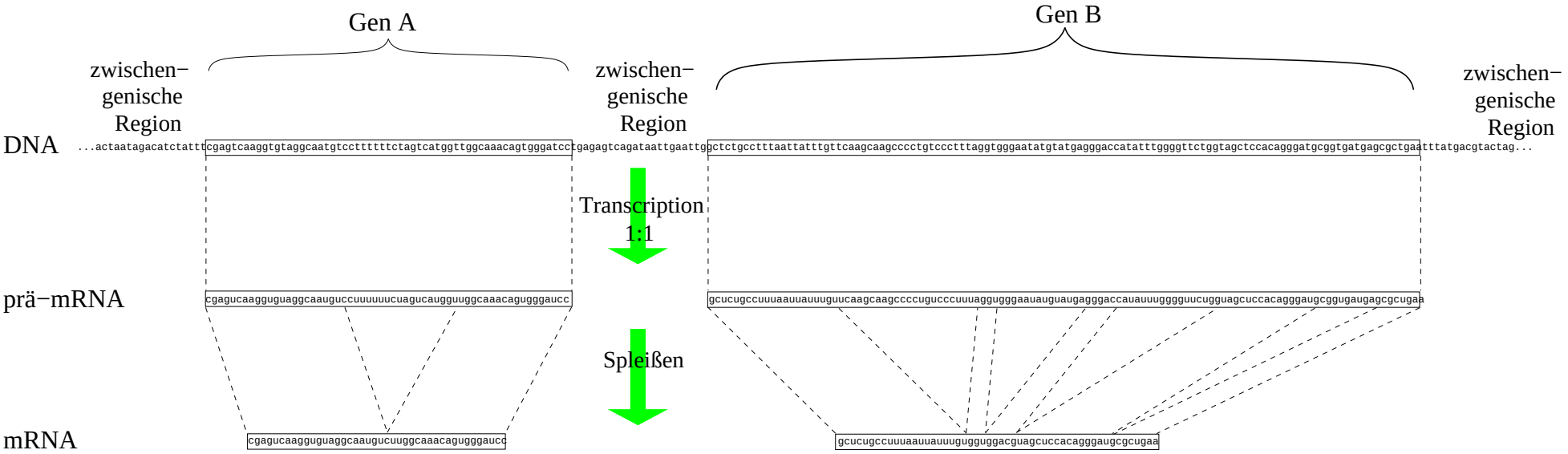
Structure of a eukaryotic gene



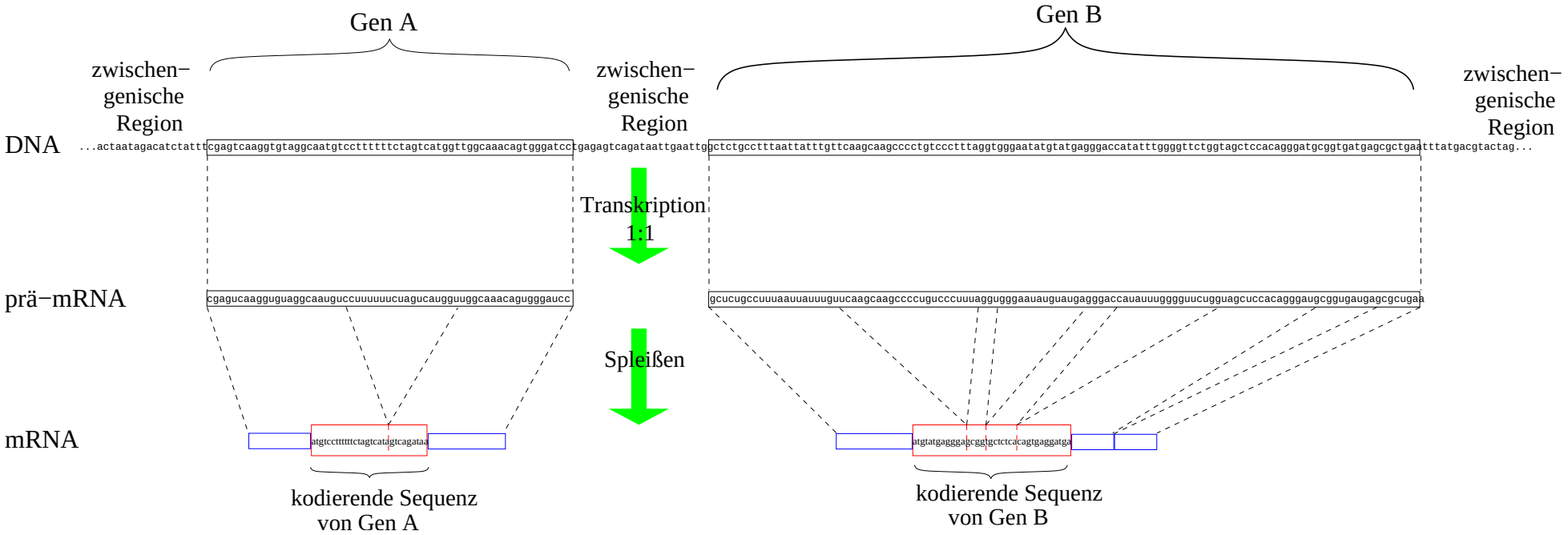
Structure of a eukaryotic gene



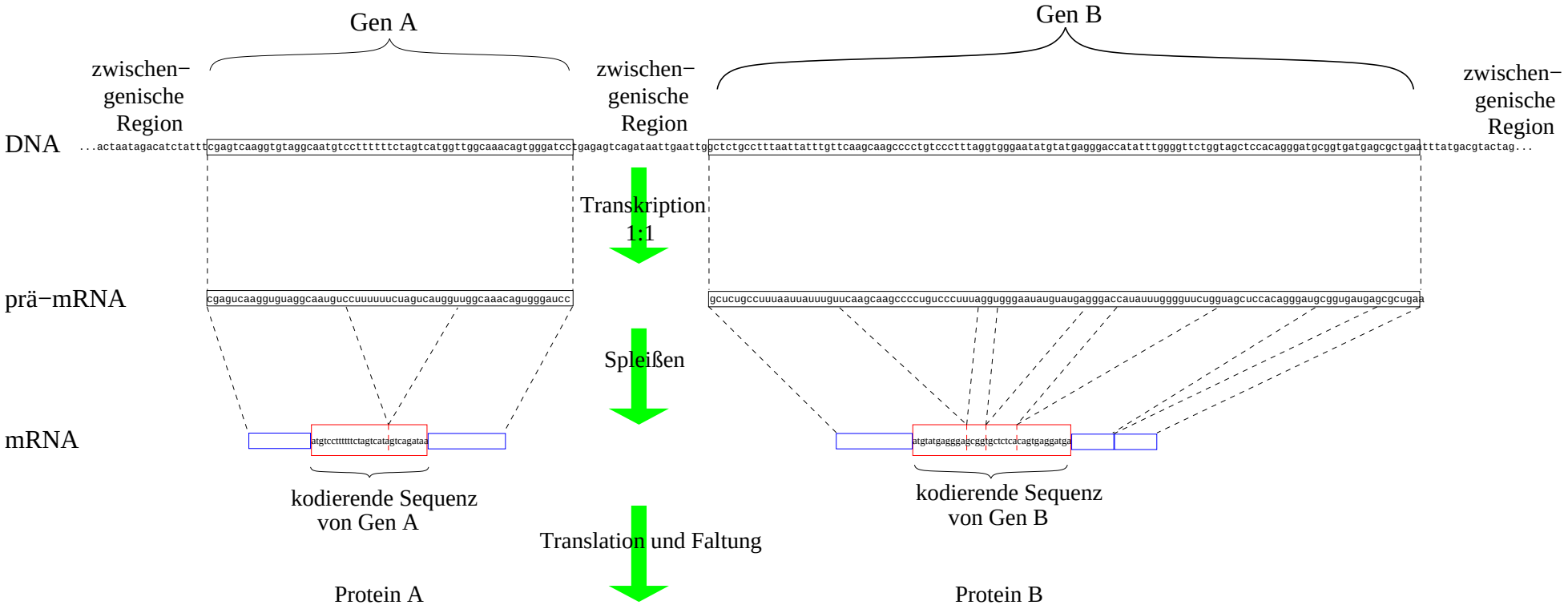
Structure of a eukaryotic gene



Structure of a eukaryotic gene



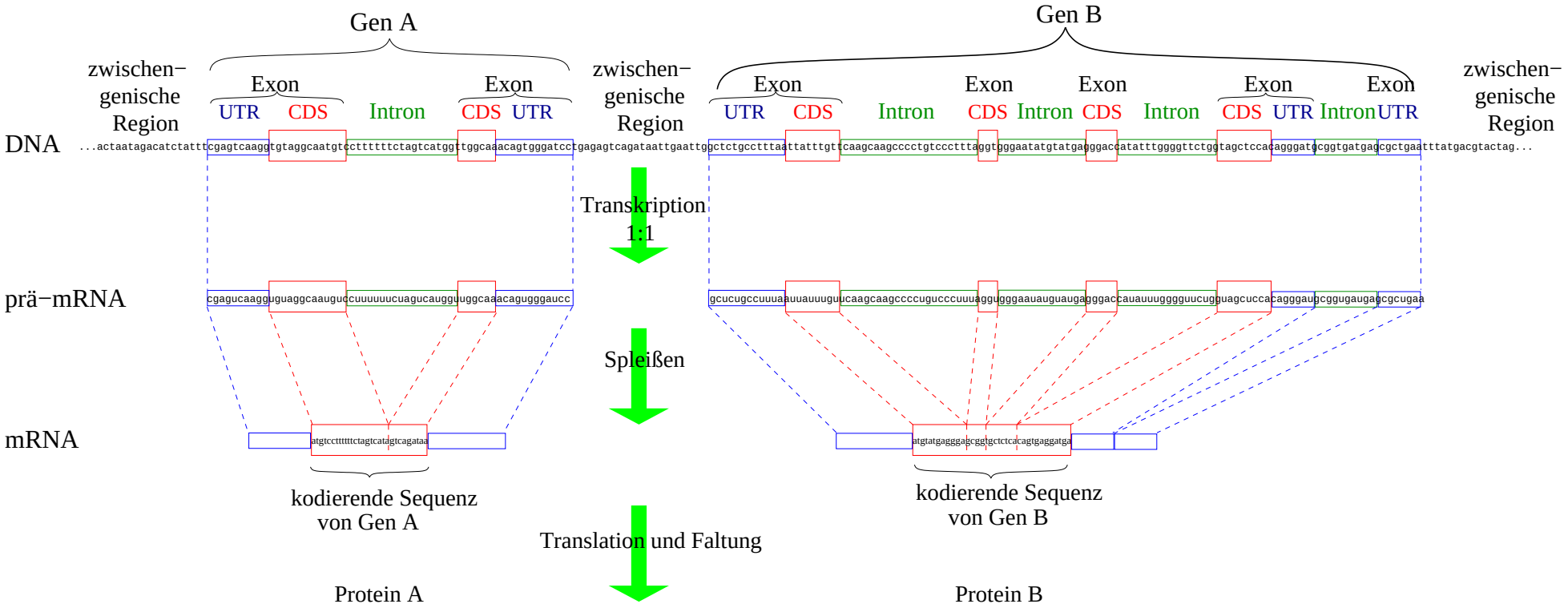
Structure of a eukaryotic gene



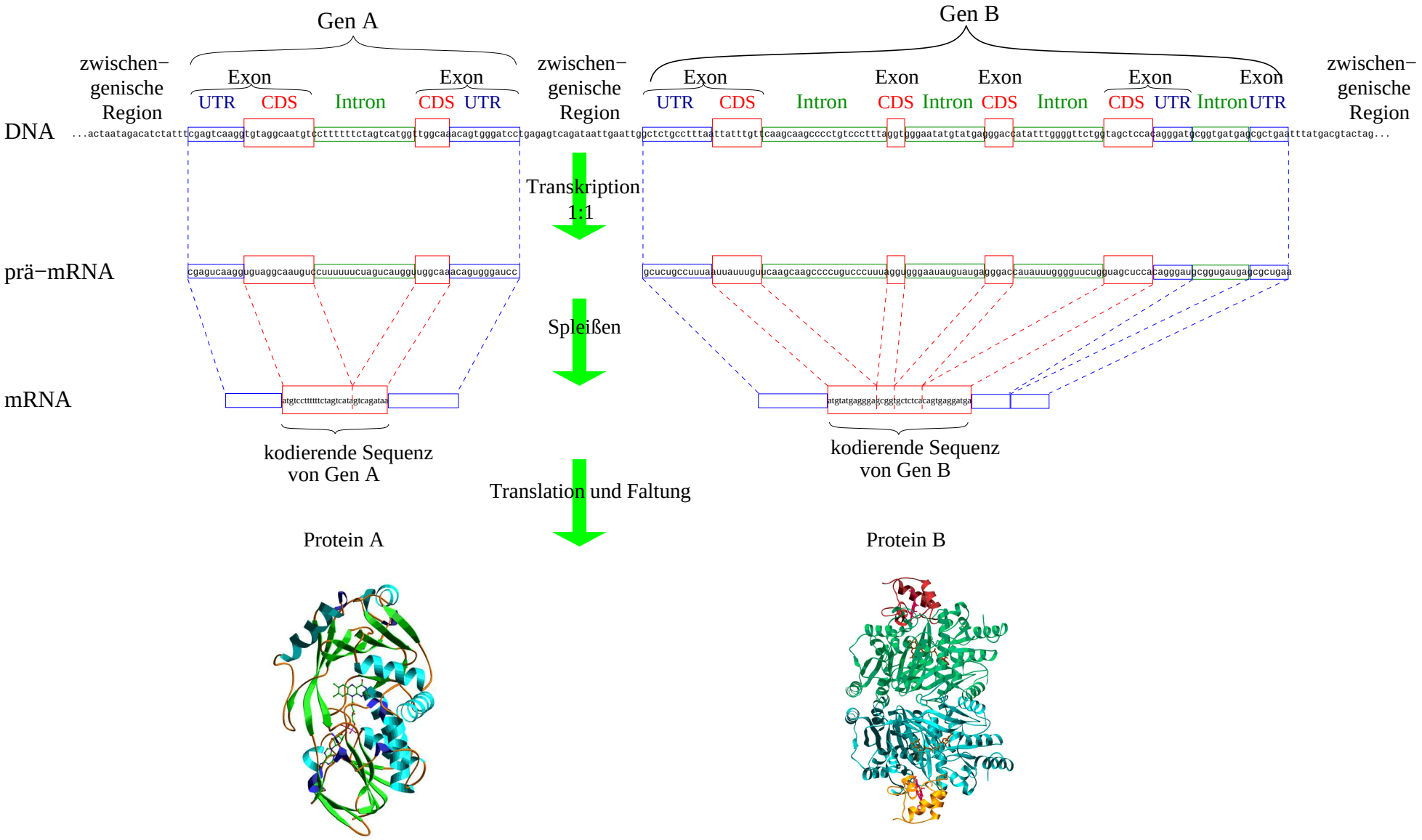
Structure of a eukaryotic gene

UTR = UnTranslated Region = part of mRNA that is not translated

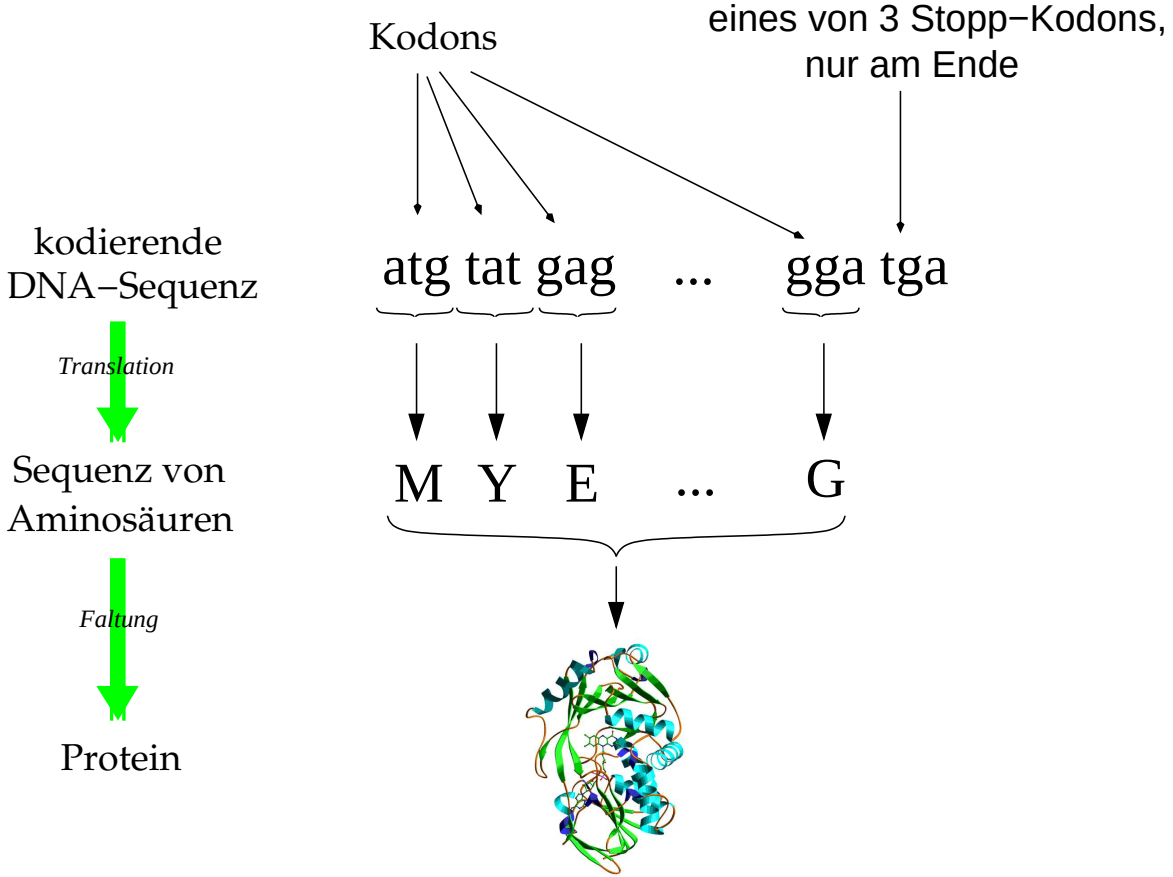
CDS = CcoDing Sequence = part of mRNA (exon) that is translated



Structure of a eukaryotic gene



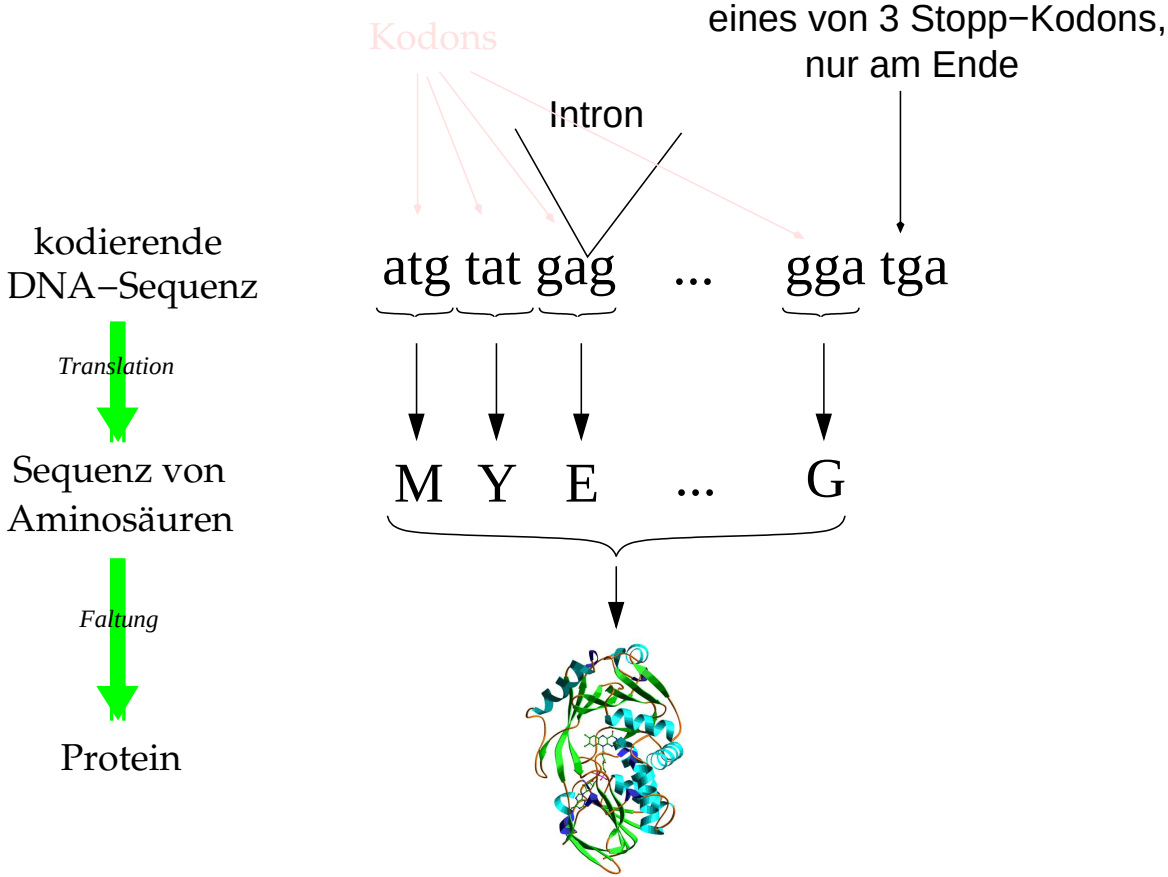
Translation



"universeller" genetischer Code

Kodon (DNA)		Amino-säure
aaa	↦	K
aac	↦	N
aag	↦	K
aat	↦	N
.		.
.		.
.		.
atg	↦	M
.		.
.		.
.		.
61 Kodons		20 Amino-säuren

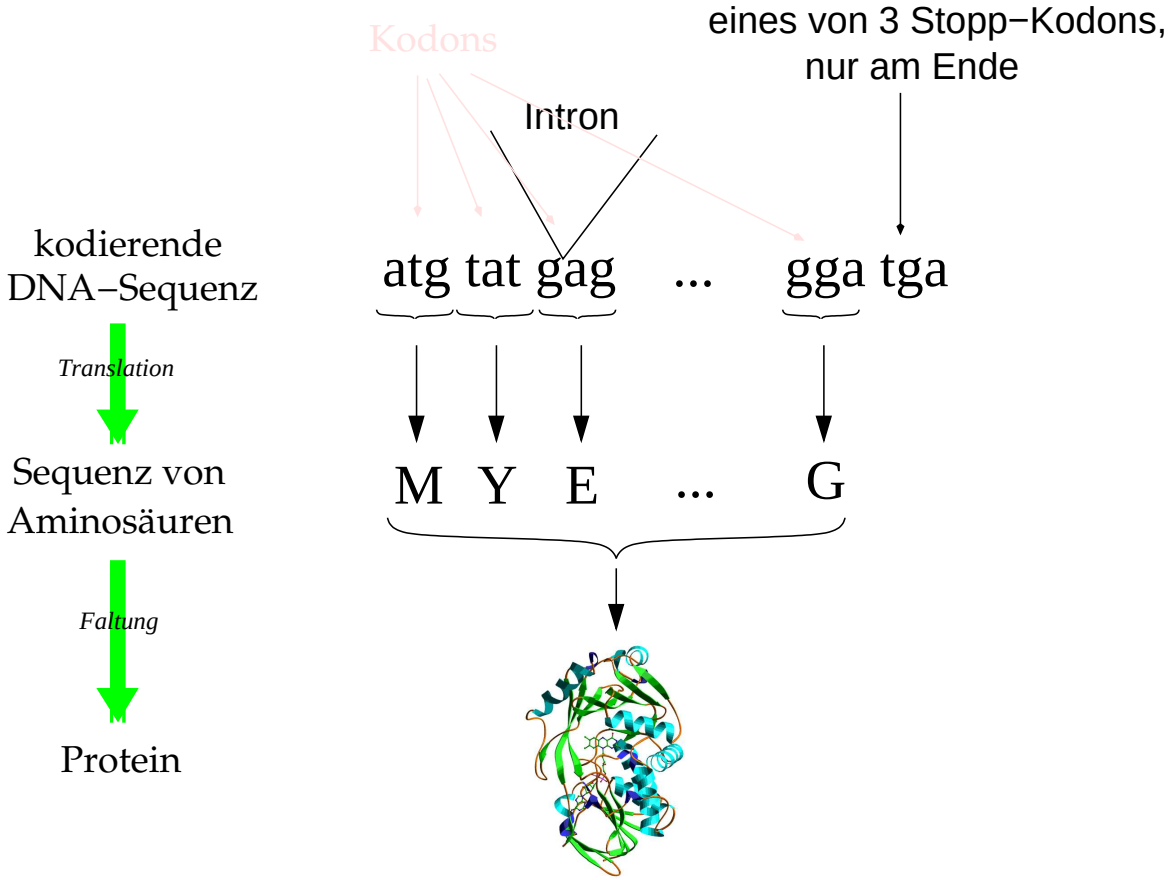
Translation



"universeller" genetischer Code

Kodon (DNA)		Amino-säure
aaa	↦	K
aac	↦	N
aag	↦	K
aat	↦	N
.		.
.		.
.		.
atg	↦	M
.		.
.		.
.		.
61 Kodons		20 Amino-säuren

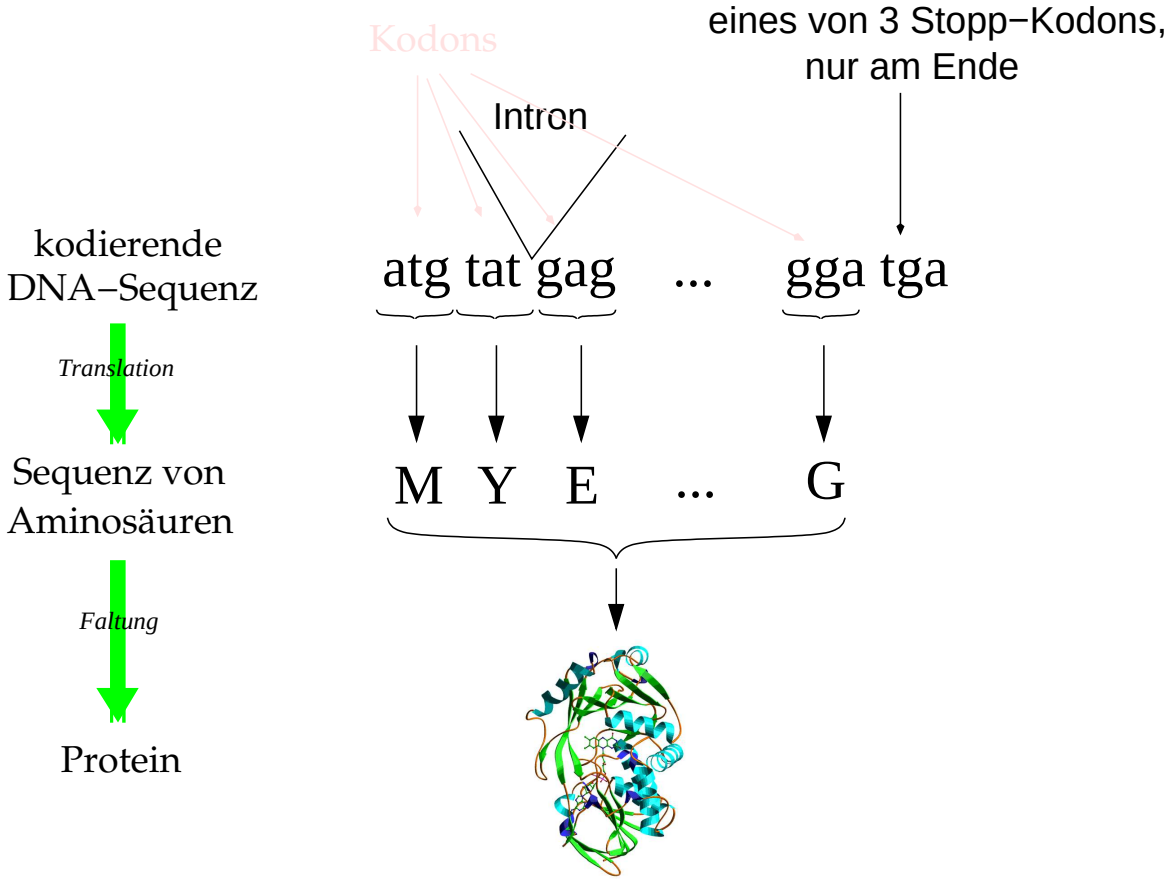
Translation



"universeller" genetischer Code

Kodon (DNA)		Amino-säure
aaa	↦	K
aac	↦	N
aag	↦	K
aat	↦	N
.		.
.		.
.		.
atg	↦	M
.		.
.		.
.		.
61 Kodons		20 Amino-säuren

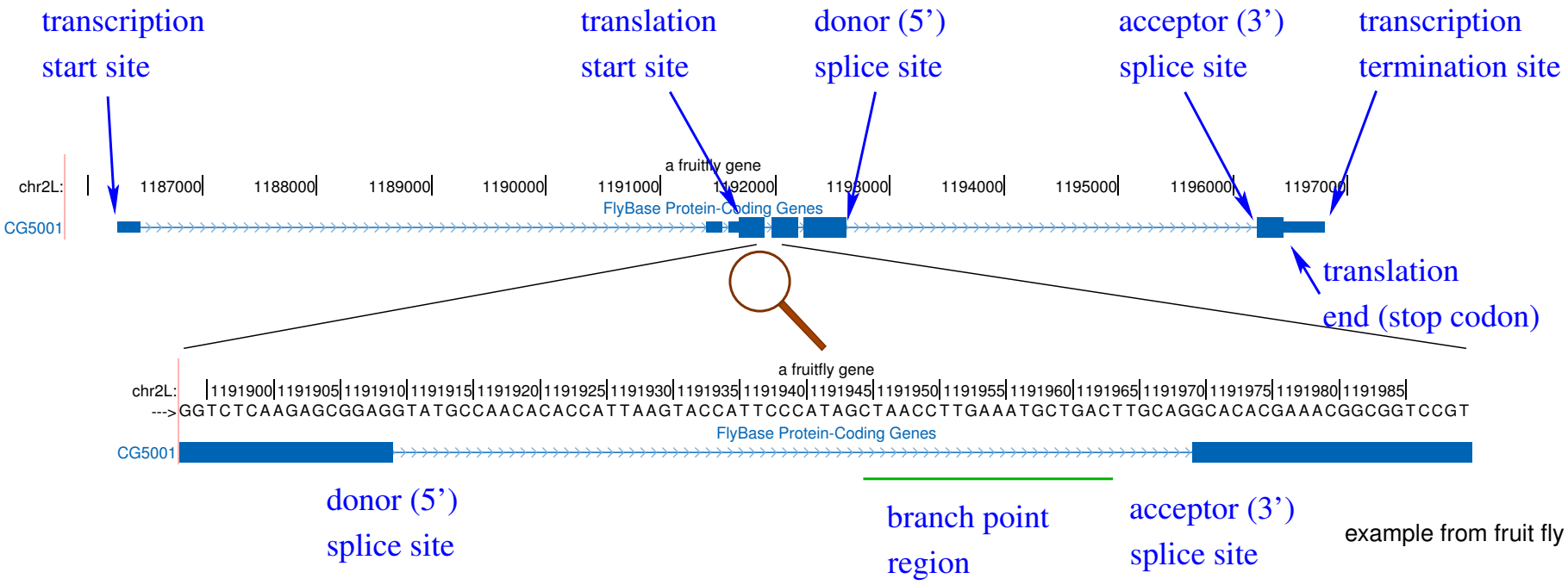
Translation



"universeller" genetischer Code

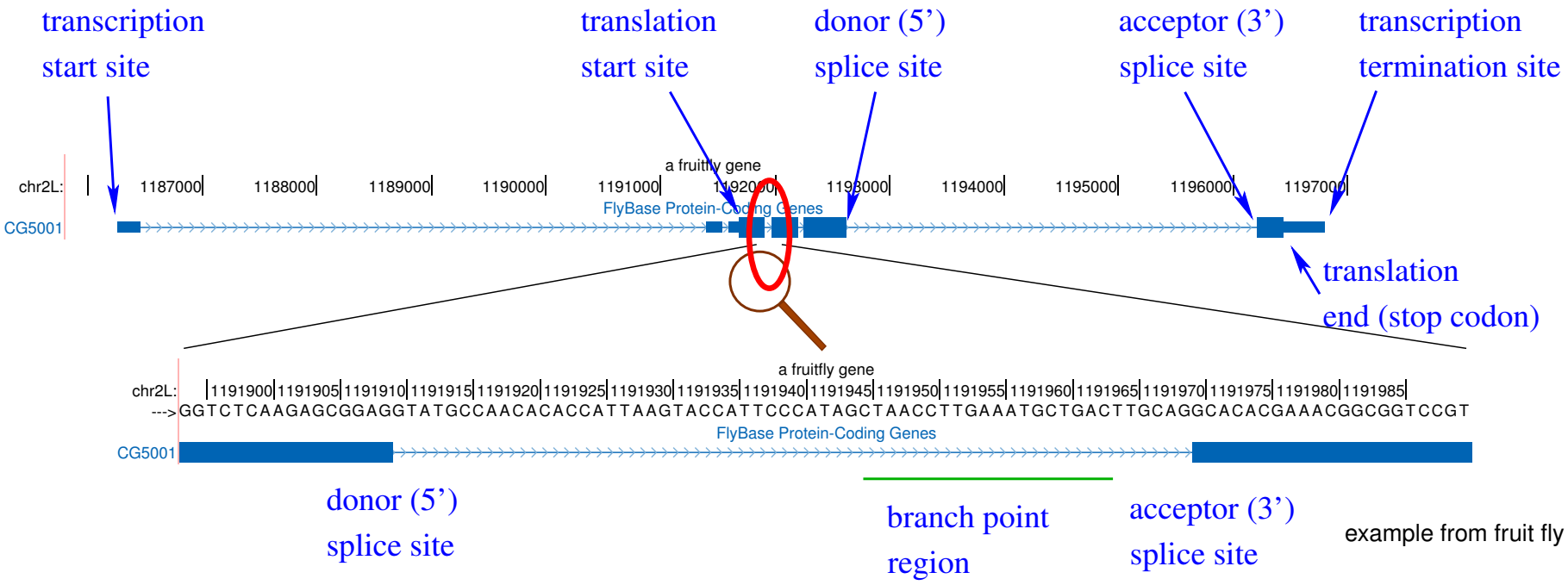
Kodon (DNA)		Amino-säure
aaa	↦	K
aac	↦	N
aag	↦	K
aat	↦	N
.		.
.		.
.		.
atg	↦	M
.		.
.		.
.		.
61 Kodons		20 Amino-säuren

Signals

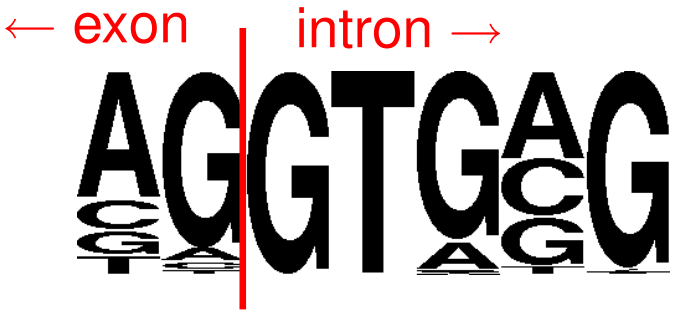


example from fruit fly

Signals



example from fruit fly



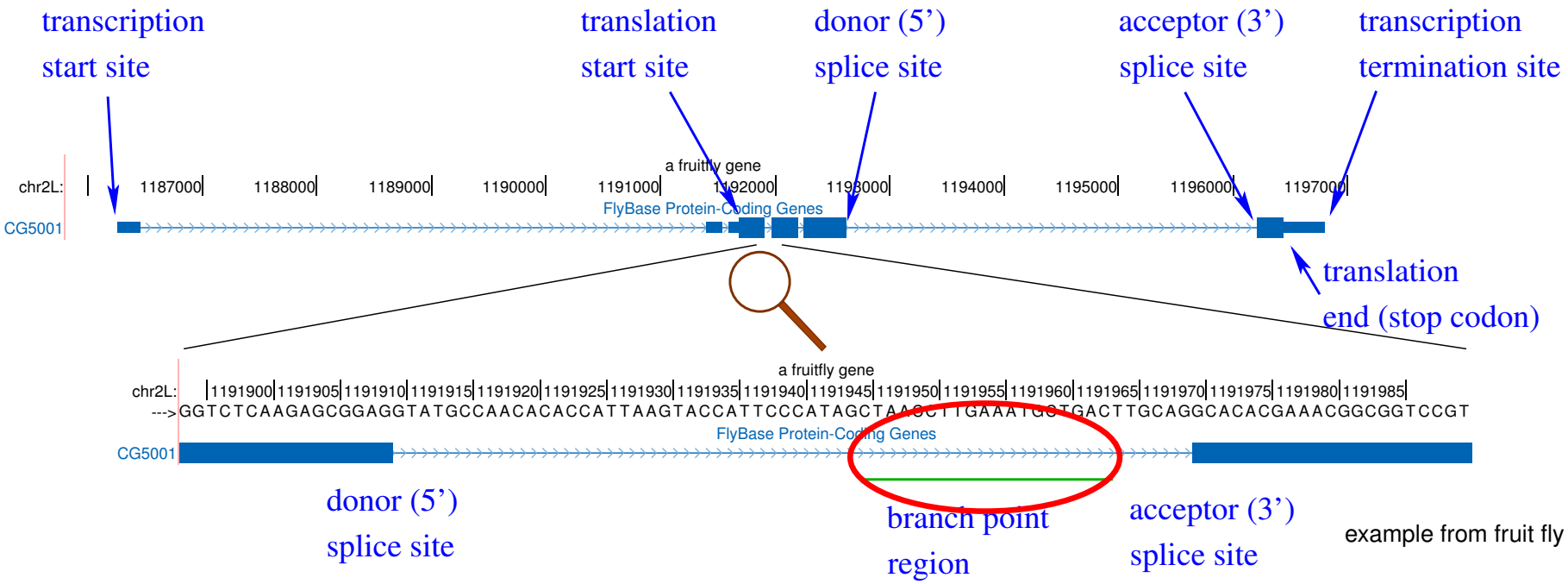
donor splice site (DSS) signal

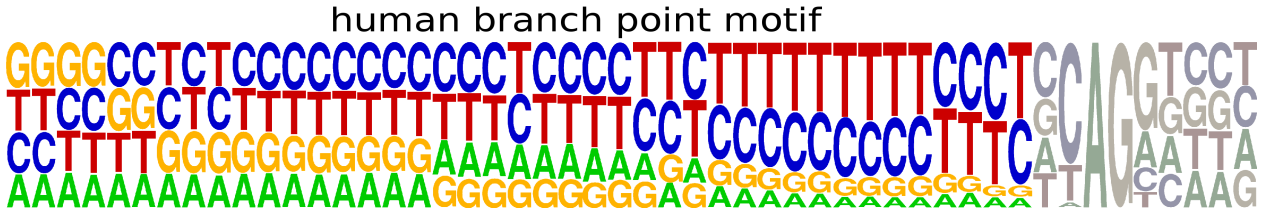
acceptor splice site (ASS) signal

Frequency of the nucleotides at positions relative to splice site.

from green algae *Chlamydomonas*

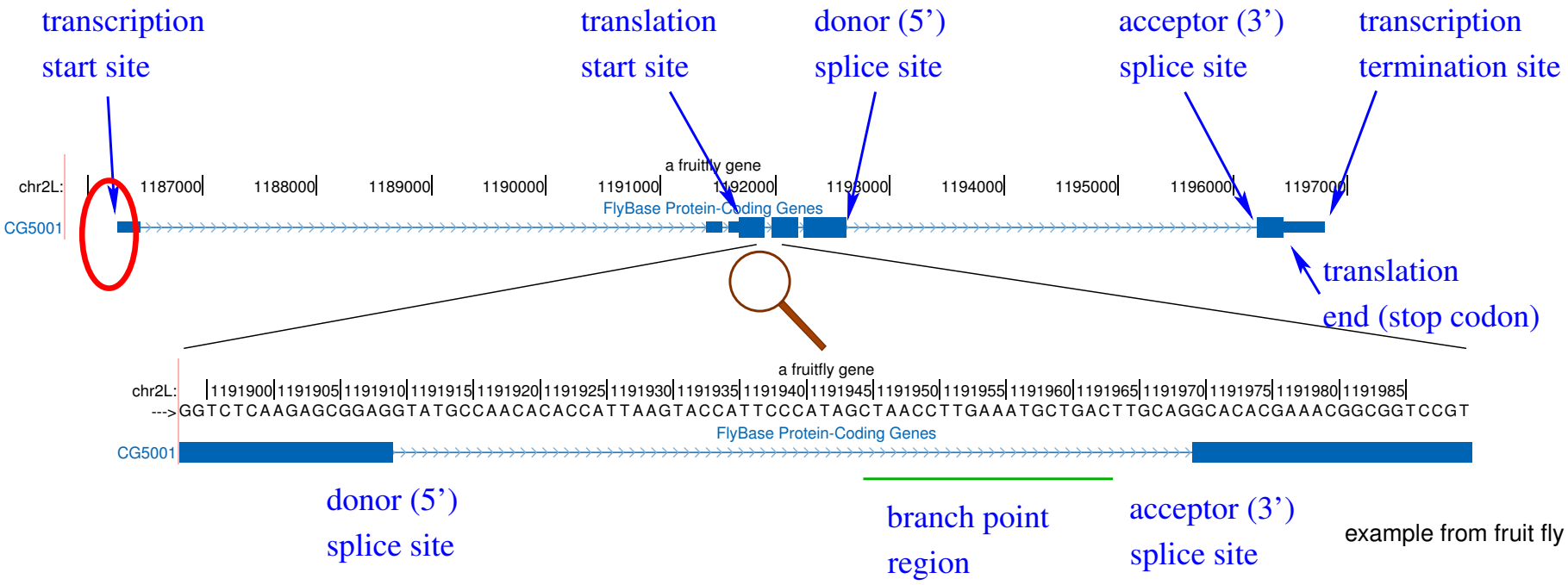
Signals





Branch point: upstream of 3' splice site, a single conserved adenine at variable distance to 3' splice site ($\approx$ -30), a splicing complex binds to it, pyrimidine (C,T) rich in human

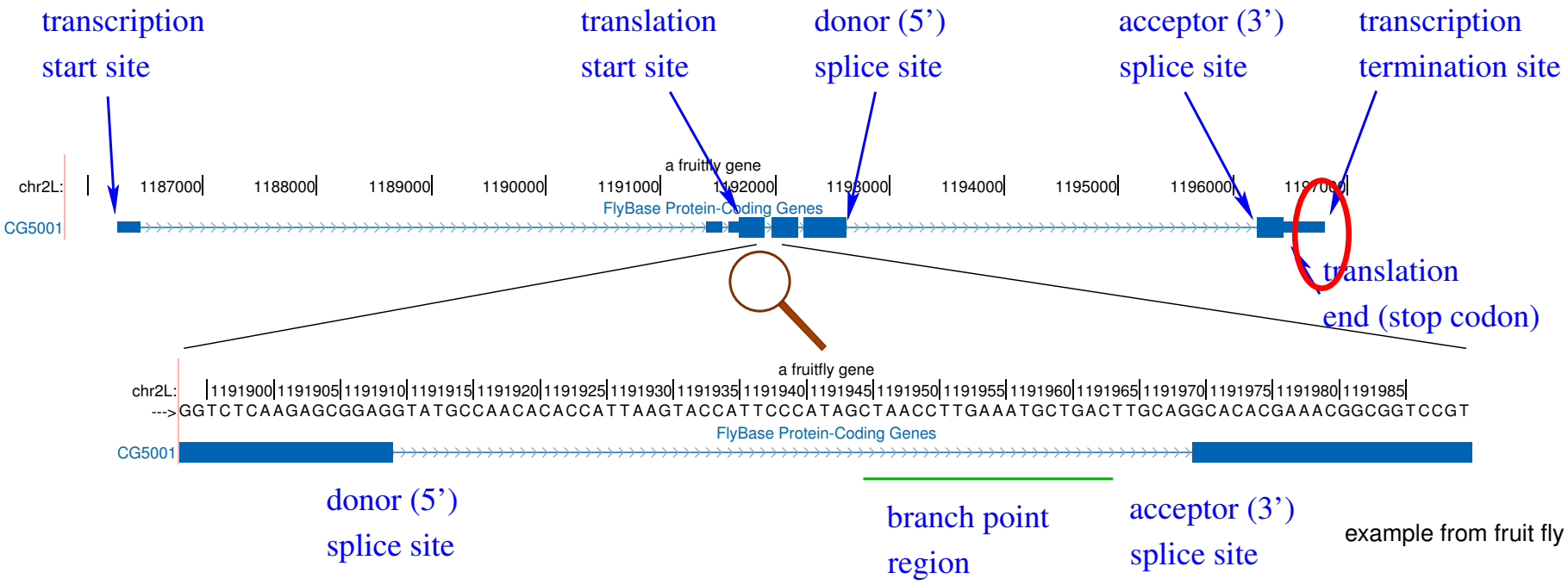
Signals



Transcription start site: Transcription from DNA to RNA by RNA polymerase starts here facilitated by promoter elements.
Promoter elements are diverse and their profiles tend to contain little info:

- diverse transcription factor binding sites at very variable positions
- sometimes TATA-box
- "CpG islands"

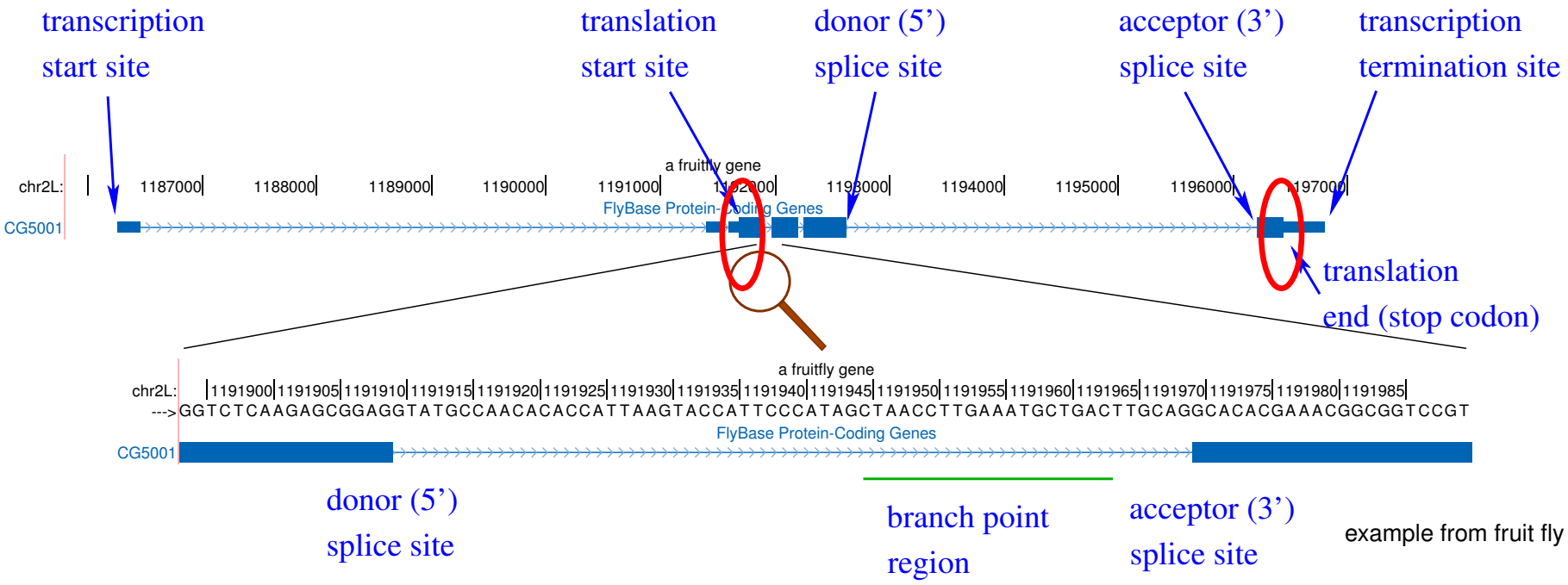
Signals



Transcription termination site (TTS):

- cleavage of the transcript.
- some non-templated A's are appended (polyadenylation).
- polyadenylation is triggered in many species in many genes by the hexamer aataaa roughly 15 bp upstream of the TTS.

Signals



Start and stop codon:

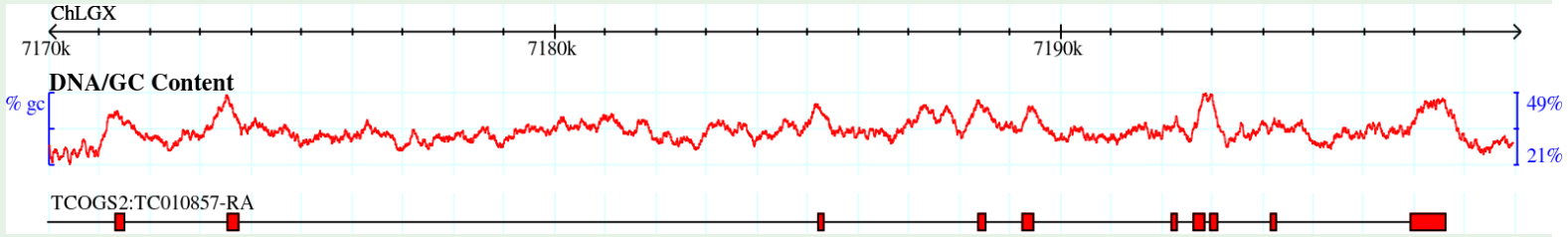
- start codon: ATG
- stop codons: TAA, TAG, TGA
  In some species the genetic code is altered and a "stop codon" is actually coding for an amino acid.

Nucleotide Composition of Coding and Noncoding Regions

Sequence Content

Besides the signals, position-unspecific frequencies of nucleotide patterns can be used to guess biological classification (e.g. CDS, non-coding, CpG-island) of longer sequence intervals.

Example (GC content in red flour beetle)



Typically, higher order patterns are examined:
E.g. reading-frame dependent k-mer frequencies ($k = 5, 6$) for protein-coding regions.

Remark

Sequence content is usually only indirect evidence.

Problems and General Ansatz

Problems

- known signal models do not carry much information

Problems and General Ansatz

Problems

- known signal models do not carry much information
- false positive signals because of low number of true positives

Problems and General Ansatz

Problems

- known signal models do not carry much information
- false positive signals because of low number of true positives
- sequence content can be misleading (pseudogenes, repeats)

Problems and General Ansatz

Problems

- known signal models do not carry much information
- false positive signals because of low number of true positives
- sequence content can be misleading (pseudogenes, repeats)

Ansatz

- combine all individual weak info to boost discriminatory power

Problems and General Ansatz

Problems

- known signal models do not carry much information
- false positive signals because of low number of true positives
- sequence content can be misleading (pseudogenes, repeats)

Ansatz

- combine all individual weak info to boost discriminatory power
- enforce standard gene structure:
  - reading frame consistency between exons
  - minimal splice site consensus (GT/AG, maybe GC/AG)
  - no in-frame stop codons
  - minimal intron length ($\approx$ 40 bp)

This Section Also in My German Script

http://gobics.de/mario/genomanalyse/script.pdf
pages 28-32

Problem Definition

Definition

Let $\mathcal{B} = \{B_1, B_2, \ldots, B_n\}$ be a set of intervals with boundaries given by $B_j = [\ell_j, r_j)$ and $\ell_j < r_j$, $(j = 1, \ldots, n)$.
Let $s_j \in \mathbb{R}$ be the score of interval B_j.

Problem Definition

Definition

Let $\mathcal{B} = \{B_1, B_2, \ldots, B_n\}$ be a set of intervals with boundaries given by $B_j = [\ell_j, r_j)$ and $\ell_j < r_j$, $(j = 1, \ldots, n)$.
Let $s_j \in \mathbb{R}$ be the score of interval B_j.
A chain $\Gamma = (B_{j_1}, B_{j_2}, \ldots, B_{j_d})$ is a sorted sequence of non-overlapping intervals (i.e. $r_{j_i} \leq \ell_{j_{i+1}}$).

Problem Definition

Definition

Let $\mathcal{B} = \{B_1, B_2, \ldots, B_n\}$ be a set of intervals with boundaries given by $B_j = [\ell_j, r_j)$ and $\ell_j < r_j$, $(j = 1, \ldots, n)$.
Let $s_j \in \mathbb{R}$ be the score of interval B_j.
A chain $\Gamma = (B_{j_1}, B_{j_2}, \ldots, B_{j_d})$ is a sorted sequence of non-overlapping intervals (i.e. $r_{j_i} \leq \ell_{j_{i+1}}$).
The score of a chain is the sum of the scores of its intervals: $s(\Gamma) = \sum_i^d s_{j_i}$

Problem Definition

Definition

Let $\mathcal{B} = \{B_1, B_2, \ldots, B_n\}$ be a set of intervals with boundaries given by $B_j = [\ell_j, r_j)$ and $\ell_j < r_j$, $(j = 1, \ldots, n)$.
Let $s_j \in \mathbb{R}$ be the score of interval B_j.
A chain $\Gamma = (B_{j_1}, B_{j_2}, \ldots, B_{j_d})$ is a sorted sequence of non-overlapping intervals (i.e. $r_{j_i} \leq \ell_{j_{i+1}}$).
The score of a chain is the sum of the scores of its intervals: $s(\Gamma) = \sum_i^d s_{j_i}$

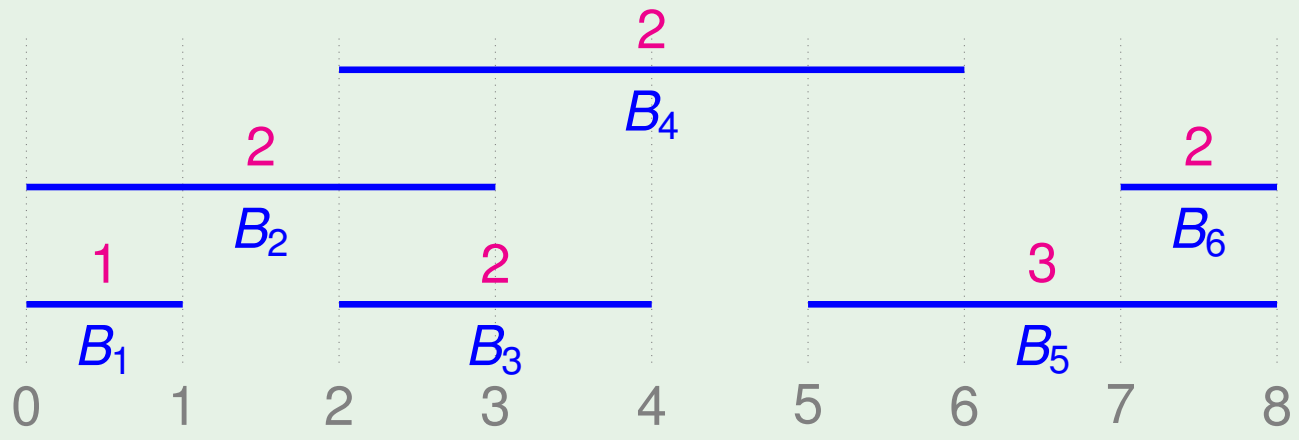
Definition (One-dimensional Chaining Problem)

For a given set of scored intervals $\mathcal{B}$ find a chain with maximal score.

Example Chaining Problem

Example

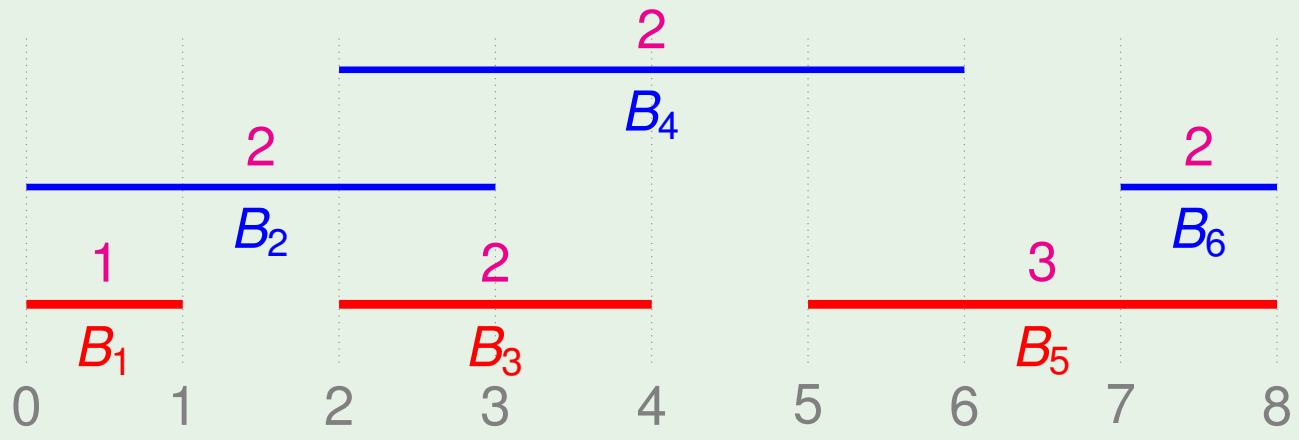
$B_1 = [0, 1), s_1 = 1$
$B_2 = [0, 3), s_2 = 2$
$B_3 = [2, 4), s_3 = 2$
$B_4 = [2, 6), s_4 = 2$
$B_5 = [5, 8), s_5 = 3$
$B_6 = [7, 8), s_6 = 2$
$\mathcal{B} = \{B_1, \ldots, B_6\}$



Example Chaining Problem

Example

$B_1 = [0, 1), s_1 = 1$
$B_2 = [0, 3), s_2 = 2$
$B_3 = [2, 4), s_3 = 2$
$B_4 = [2, 6), s_4 = 2$
$B_5 = [5, 8), s_5 = 3$
$B_6 = [7, 8), s_6 = 2$
$\mathcal{B} = \{B_1, \ldots, B_6\}$



$\Gamma = (B_1, B_3, B_5)$ is *the* chain with maximal score.

How to Solve the Chaining Problem?

- brute force too slow: There are 2^n possible chains.

How to Solve the Chaining Problem?

- brute force too slow: There are 2^n possible chains.
- greedy aproach does not correctly solve the problem:

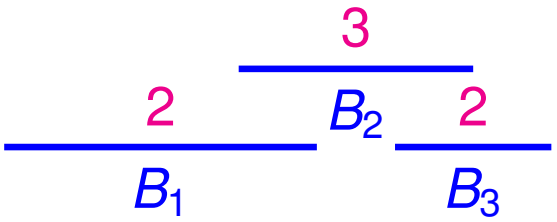
$\Gamma \leftarrow ()$
repeat
  insert highest-scoring interval into Γ that does not overlap any interval already in Γ
until no more interval can be inserted

How to Solve the Chaining Problem?

- brute force too slow: There are 2^n possible chains.
- greedy aproach does not correctly solve the problem:

$\Gamma \leftarrow ()$
repeat
  insert highest-scoring interval into Γ that does not overlap any interval already in Γ
until no more interval can be inserted

trivial counterexample:



Chaining Algorithm

One-Dimensional Chaining Algorithm

$P \leftarrow$ sort $\{\ell_1, r_1, \ell_2, r_2, \ldots, \ell_n, r_n\}$ increasingly
$S \leftarrow q \leftarrow q_1 \leftarrow \cdots \leftarrow q_n \leftarrow S_1 \leftarrow \cdots S_n \leftarrow 0$
while P not empty **do**
  $b \leftarrow$ remove smallest element in P
  for all j such that $r_j = b$ **do**
    if $S_j > S$ **then**
      $S \leftarrow S_j$
      $q \leftarrow j$
    end if
  end for
  for all j such that $\ell_j = b$ **do**
    $S_j \leftarrow s_j + S$
    $q_j \leftarrow q$
  end for
end while
output S as score of best chain

Chaining Algorithm

Backtracking

$\Gamma \leftarrow ()$
while $q \neq 0$ **do**
    push B_q onto Γ
    $q \leftarrow q_q$
end while
reverse order of Γ
output Γ as highest scoring chain

Correctness

Invariants of the Algorithm

1. After very iteration of the main loop in line 3, S is the score of the best chain without interval boundaries beyond b.
2. After every iteration of the main loop in line 3, S_j is the score of the best chain, that ends with interval B_j for all j with $\ell_j \leq b$.

Proof by induction on the iteration of the main loop in line 3.
It follows that after the last iteration S is the score of the overall best chain.

Pointers for Backtracking

Unless undefined ($q_j = 0$), q_j is the index of the interval immediately left of B_j in a best chain that contains B_j.

Example Algorithm Run

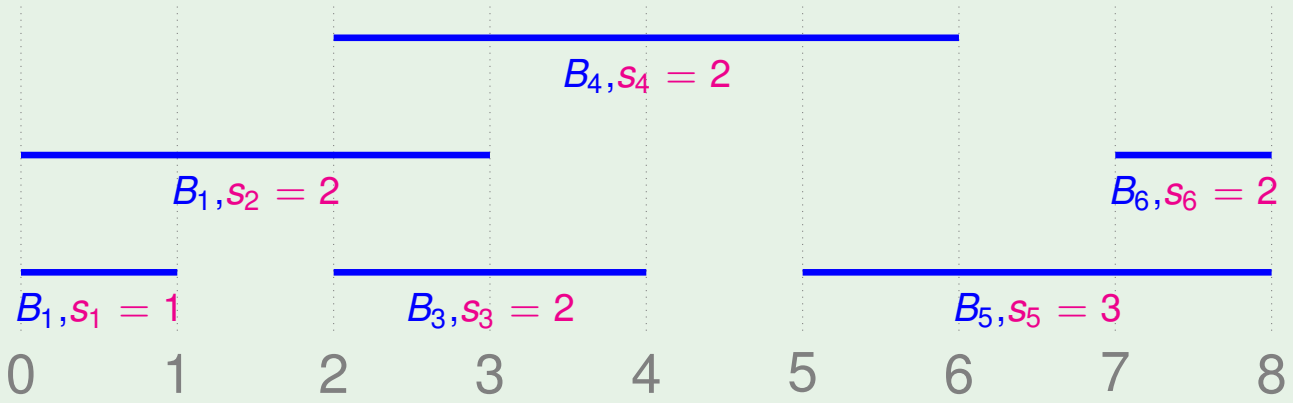
Example

After initialization (line 2):

$P = (0, 1, 2, 3, 4, 5, 6, 7, 8)$

$S = 0$

$q = 0$



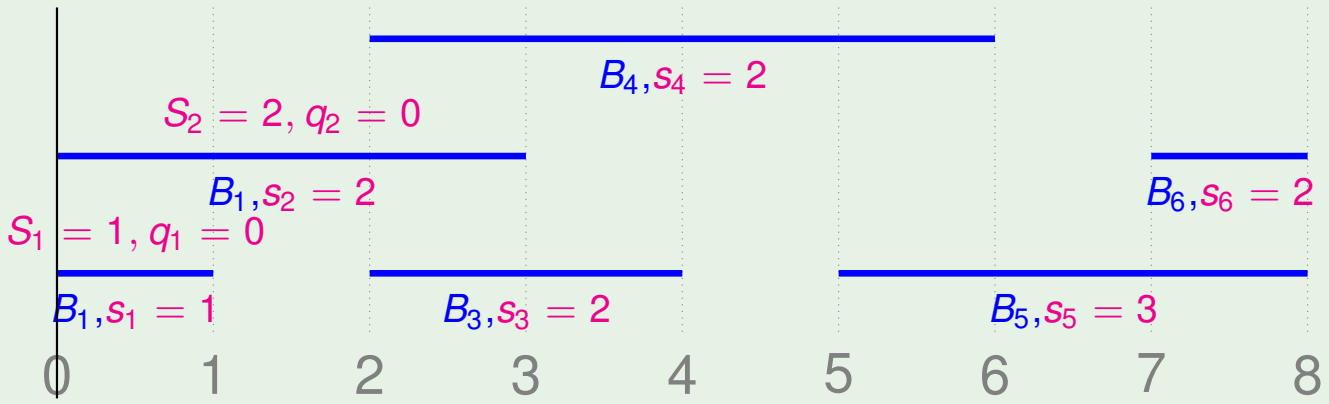
Example Algorithm Run

Example

After 1st iteration of main loop (line 3):

$S = 0$

$q = 0$



$b = 0$

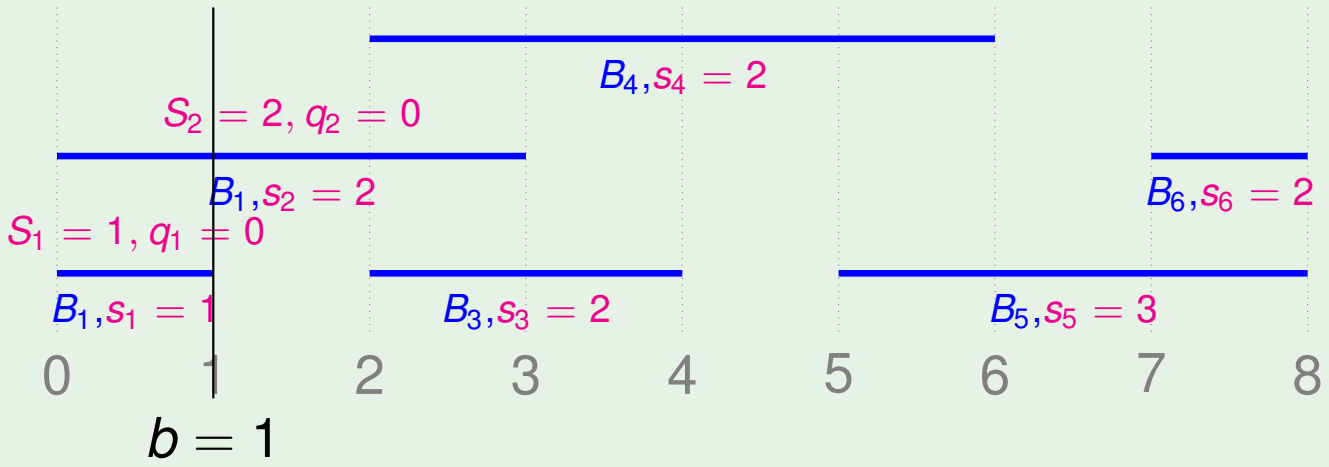
Example Algorithm Run

Example

After 2nd iteration of main loop (line 3):

$S = 1$

$q = 1$



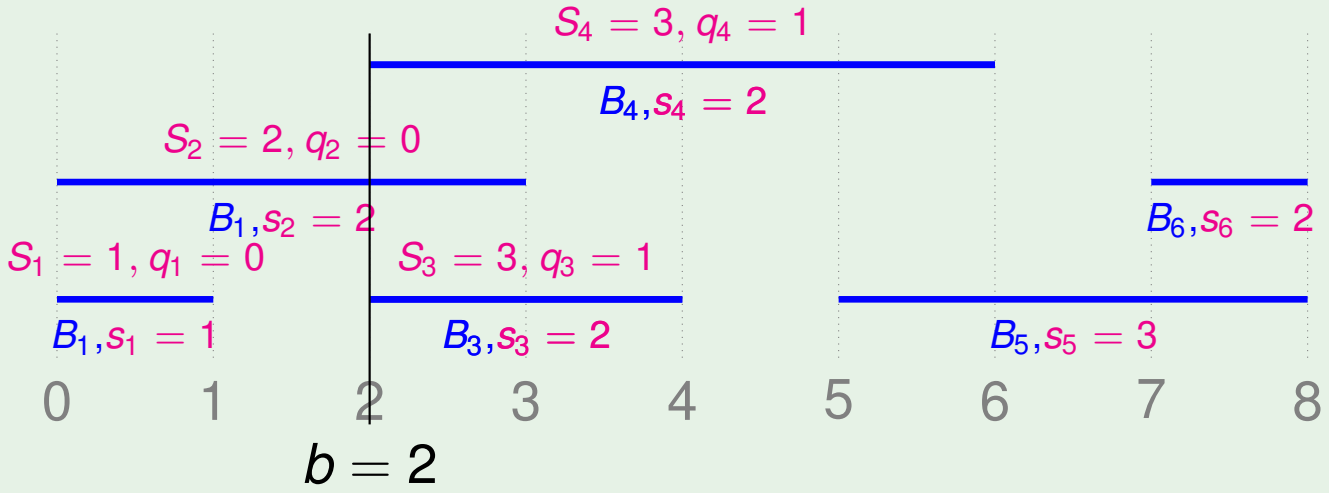
Example Algorithm Run

Example

After 3rd iteration of main loop (line 3):

$S = 1$

$q = 1$



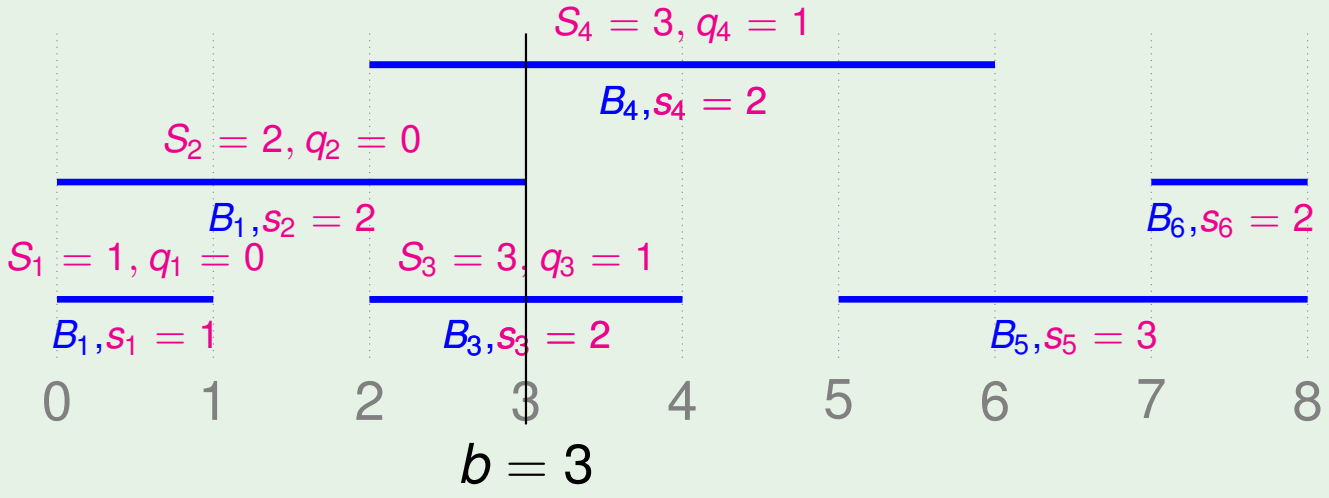
Example Algorithm Run

Example

After 4th iteration of main loop (line 3):

$S = 2$

$q = 2$



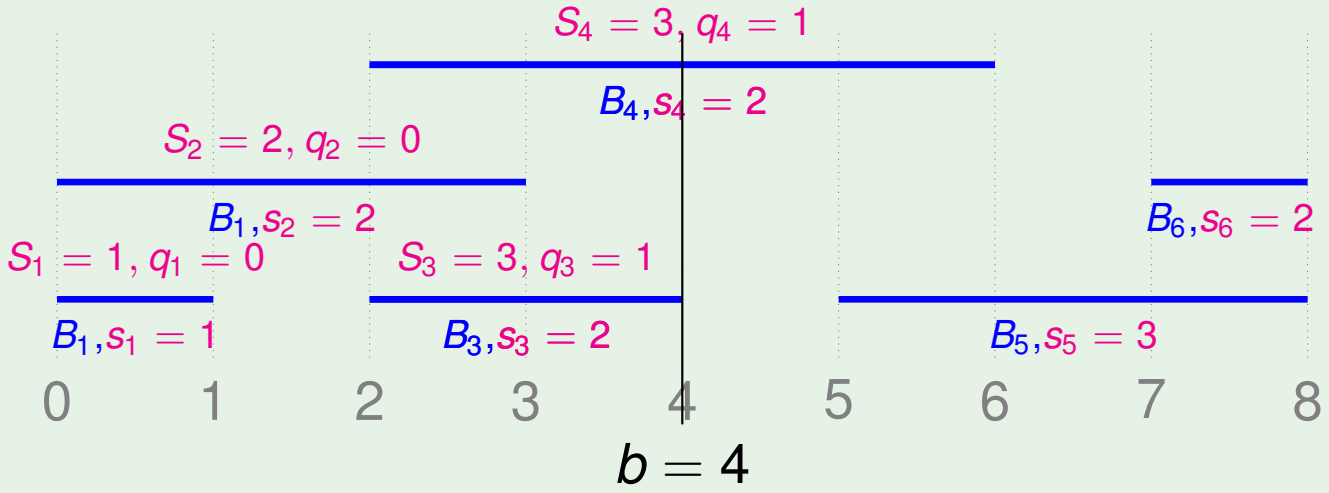
Example Algorithm Run

Example

After 5th iteration of main loop (line 3):

$S = 3$

$q = 3$



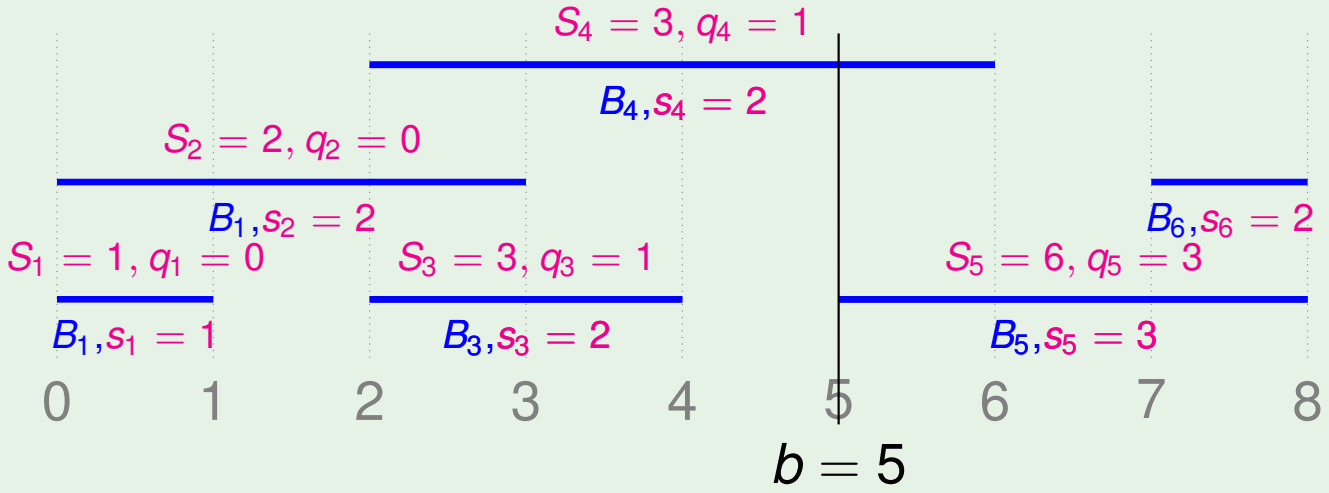
Example Algorithm Run

Example

After 6th iteration of main loop (line 3):

$S = 3$

$q = 3$



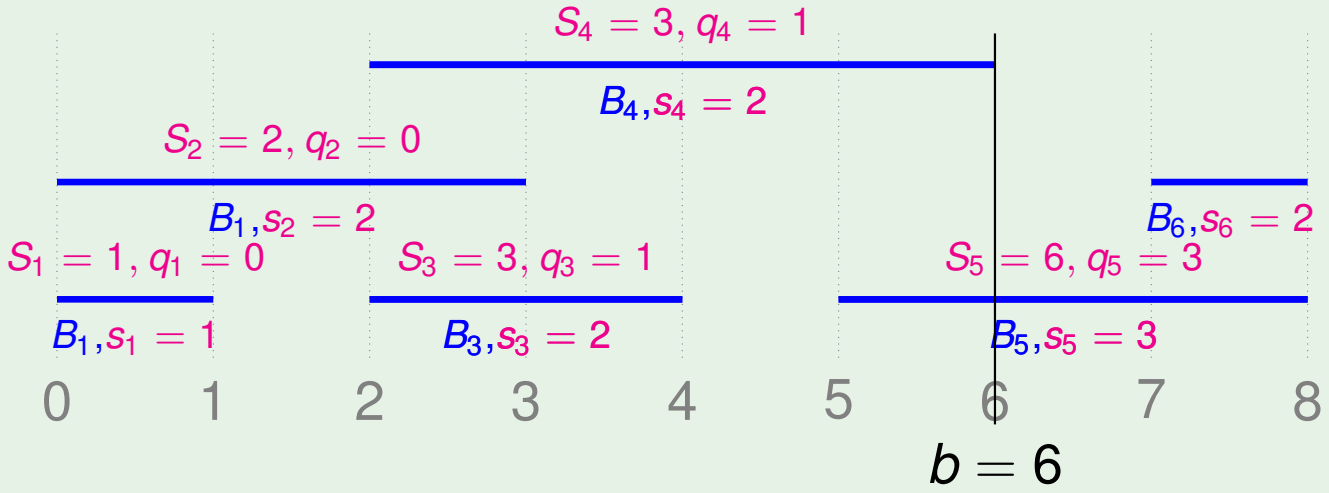
Example Algorithm Run

Example

After 7th iteration of main loop (line 3):

$S = 3$

$q = 3$



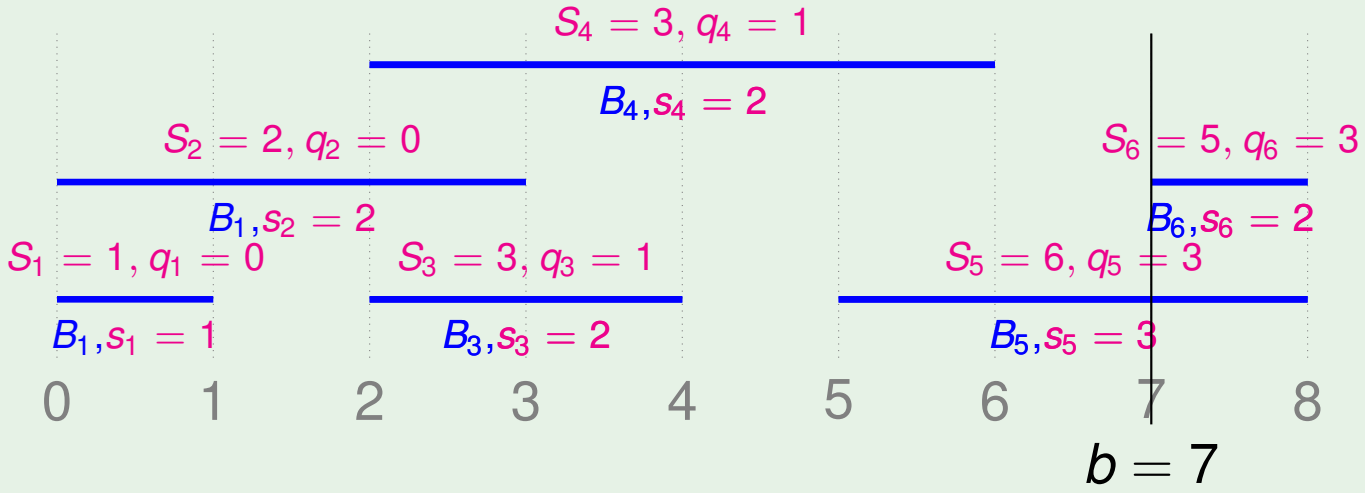
Example Algorithm Run

Example

After 8th iteration of main loop (line 3):

$S = 3$

$q = 3$



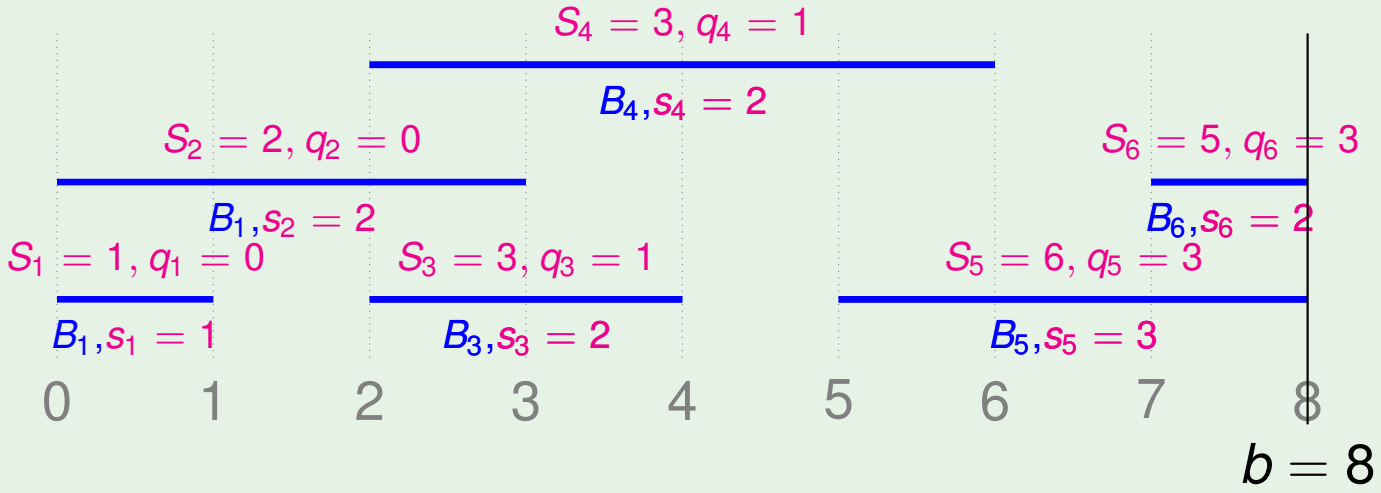
Example Algorithm Run

Example

After last iteration of main loop (line 3):

$S = 6$

$q = 5$



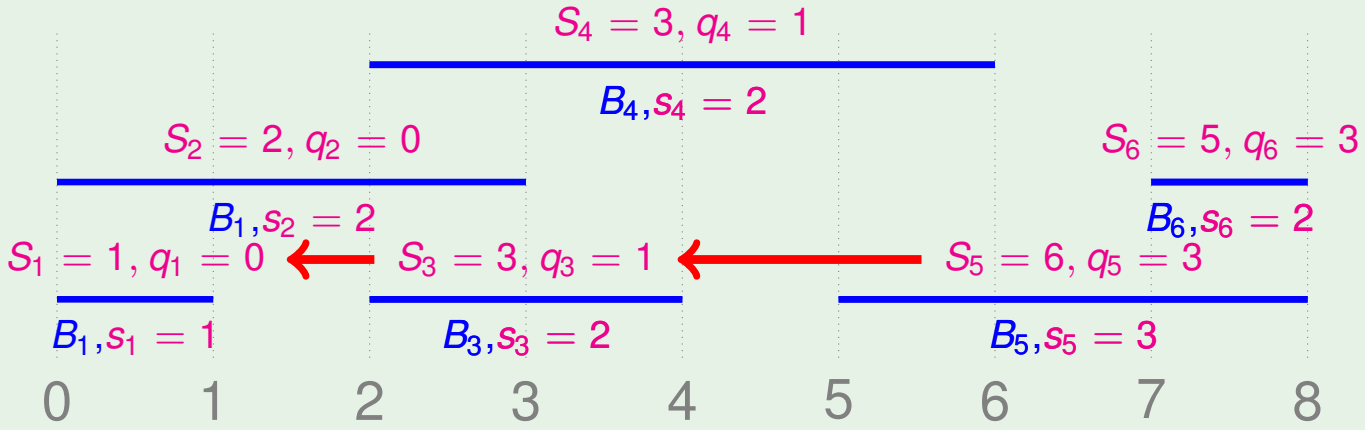
Example Algorithm Run

Example

Backtracking:

Follow q_j pointers starting from $q = 5$ until $q = 0$.

$\Gamma = (B_1, B_3, B_5)$



Running Time

Running Time

Sorting of interval boundaries (line 1):

Running Time

Running Time

Sorting of interval boundaries (line 1): $O(n \log n)$
Overall time in main loop (lines 3-15):

Running Time

Running Time

Sorting of interval boundaries (line 1): $O(n \log n)$
Overall time in main loop (lines 3-15): $O(n)$
Backtracking:

Running Time

Running Time

Sorting of interval boundaries (line 1): $O(n \log n)$
Overall time in main loop (lines 3-15): $O(n)$
Backtracking: $O(n)$
Overall running time:

Running Time

Running Time

Sorting of interval boundaries (line 1): $O(n \log n)$
Overall time in main loop (lines 3-15): $O(n)$
Backtracking: $O(n)$
Overall running time: $O(n \log n)$

Running Time

Running Time

Sorting of interval boundaries (line 1): $O(n \log n)$
Overall time in main loop (lines 3-15): $O(n)$
Backtracking: $O(n)$
Overall running time: $O(n \log n)$

Remarks:

- The linear running time of the main loop can be realized when for each interval boundary in P a list of intervals ending and starting at b is stored. For each interval the loops 5-10 and 11-14 are then executed exactly once each (amortized analysis).

Running Time

Running Time

Sorting of interval boundaries (line 1): $O(n \log n)$
Overall time in main loop (lines 3-15): $O(n)$
Backtracking: $O(n)$
Overall running time: $O(n \log n)$

Remarks:

- The linear running time of the main loop can be realized when for each interval boundary in P a list of intervals ending and starting at b is stored. For each interval the loops 5-10 and 11-14 are then executed exactly once each (amortized analysis).
- Special but important case: the intervals have integers as boundaries (sequence positions) in the range $1..t$
  $\Rightarrow$ sorting can be done in $O(t + n)$ using Bucket Sort
  $\Rightarrow$ faster if $t = o(n \log n)$ (dense intervals)

Simple Approach to Gene Finding

- only predict protein-coding part of genes (easier)

Simple Approach to Gene Finding

- only predict protein-coding part of genes (easier)
- interpret gene structure as chain of CDS

Simple Approach to Gene Finding

- only predict protein-coding part of genes (easier)
- interpret gene structure as chain of CDS
- gene boundaries are implied by CDS boundaries (stop codon)

Simple Approach to Gene Finding

- only predict protein-coding part of genes (easier)
- interpret gene structure as chain of CDS
- gene boundaries are implied by CDS boundaries (stop codon)
- CDS candidate defined by sequence (integer) interval $B_j = [\ell_j, r_j)$
  score j-th CDS candidate:

$$
\begin{aligned}
s_j \quad = \quad & \text{score of signal at } \ell_j \quad \text{(e.g. ASS or start codon)} \\
& + \text{score of signal at } r_j \quad \text{(e.g. DSS or stop codon)} \\
& + \text{score of sequence content in } [\ell_j, r_j)
\end{aligned}
$$

Simple Approach to Gene Finding

- only predict protein-coding part of genes (easier)
- interpret gene structure as chain of CDS
- gene boundaries are implied by CDS boundaries (stop codon)
- CDS candidate defined by sequence (integer) interval $B_j = [\ell_j, r_j)$
  score j-th CDS candidate:

$$\begin{aligned} s_j \quad = \quad & \text{score of signal at } \ell_j \quad \text{(e.g. ASS or start codon)} \\ & + \text{score of signal at } r_j \quad \text{(e.g. DSS or stop codon)} \\ & + \text{score of sequence content in } [\ell_j, r_j) \end{aligned}$$

- use chaining algorithm to find “best” exon chain

Simple Approach to Gene Finding

Signal Score

A number s assigned to a sequence position p that is used to decide whether the signal is present at p.
Usually: $s = s(w)$, where w is a sequence window around p.
Aims:

1. The larger the score, the more likely is it that there is a true signal.
2. $s(w)$ is "small" for positions p without the signal.

Example Signal Score

Example (DSS position weight matrix)

$p =$ candidate donor splice site position
$w =$ seq window 2 pos upstream and 5 pos downstream of DSS
Have position specific scoring matrix for DSS

$$m(i, b) \quad (i = 1, 2, \ldots, 7, b \in \text{A,C,G,T}),$$

$$m(i, \text{A}) + m(i, \text{C}) + m(i, \text{G}) + m(i, \text{T}) = 1$$

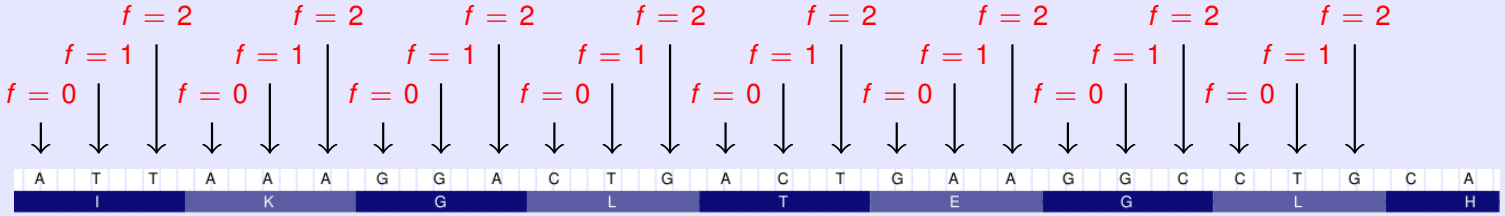
Have "background" distribution of nucleotides $q(b)$

$$q(\text{A}) + q(\text{C}) + q(\text{G}) + q(\text{T}) = 1$$

Define log-odds score: $s = \log \prod_{i=1}^{7} m(i, w_i)/q(w_i)$

Example Content Score

Base composition is frame-dependent



nucleotide frequencies in human:

	coding sequence				noncoding
	$f = 0$	$f = 1$	$f = 2$	all f	sequence
A	0.248	0.291	0.146	0.229	0.26
C	0.264	0.243	0.351	0.286	0.24
G	0.321	0.201	0.312	0.278	0.24
T	0.166	0.265	0.190	0.207	0.26

Example Content Score

Example (frame-dependent Markov chain of order k)

Let w be the DNA word of length n to be scored as CDS.
Let $f \in \{0, 1, 2\}$ be the frame of the first position of w.

$$P(w) := p_f(w_1, \ldots, w_k) \cdot \prod_{i=k+1}^{n} p_{f(i)}(w_i \mid w_{i-k}, \ldots, w_{i-1})$$

Here:
- p_f is a start probability for the first k bases
- $f(i) \in \{0, 1, 2\}$ such that $f(i) \equiv f - 1 + i \mod 3$ is the frame of the i-th position of w

Define $s(w) = \log(P(w)/Q(w))$,
where $Q(w)$ is the probability of w in a "background" model (e.g. non-coding).

Remark: division by background $\Rightarrow$ good exon candidates get positive score

Example Content Score - Continued

Example

$w = \text{ATTCTGC}$
frame $f = 2$, i.e. with these codon breaks: A||TTC||TGC
$k = 2$

$$\begin{aligned} P(\text{ATTCTGC}) &= p_2(\text{AT})p_1(\text{T}|\text{AT})p_2(\text{C}|\text{TT}) \\ &\quad p_0(\text{T}|\text{TC})p_1(\text{G}|\text{CT})p_2(\text{C}|\text{TG}) \end{aligned}$$

- if $k \geq 2$ above content model can reflect codon usage
- typical: $k = 4$ or $k = 5$
- probabilities $p_r(x \mid y_1, \ldots, y_k)$ can be estimated on known coding sequences

Problems with Simple Approach

- reading frame consistency not enforced

Problems with Simple Approach

- reading frame consistency not enforced
- $\Rightarrow$ output can be biologically “senseless”

Problems with Simple Approach

- reading frame consistency not enforced
- $\Rightarrow$ output can be biologically “senseless”
- $\Rightarrow$ less accurate when this info is ignored

Problems with Simple Approach

- reading frame consistency not enforced
- $\Rightarrow$ output can be biologically “senseless”
- $\Rightarrow$ less accurate when this info is ignored
- CDS candidates with negative score are never used

Problems with Simple Approach

- reading frame consistency not enforced
- $\Rightarrow$ output can be biologically “senseless”
- $\Rightarrow$ less accurate when this info is ignored
- CDS candidates with negative score are never used

Need extension to chaining algorithm to enforce consistency.

Consistent Chaining Problem

Definition

Let $\mathcal{B} = \{B_1, B_2, \ldots, B_n\}$ and $s_1, \ldots, s_n$ be as above.
In addition, let T be a finite set of types.

Consistent Chaining Problem

Definition

Let $\mathcal{B} = \{B_1, B_2, \ldots, B_n\}$ and $s_1, \ldots, s_n$ be as above.
In addition, let T be a finite set of types.
For every interval B_j let $\text{pre}(j), \text{suc}(j) \in T$ be a predecessor and successor type of interval j.

Consistent Chaining Problem

Definition

Let $\mathcal{B} = \{B_1, B_2, \ldots, B_n\}$ and $s_1, \ldots, s_n$ be as above.
In addition, let T be a finite set of types.
For every interval B_j let $\text{pre}(j), \text{suc}(j) \in T$ be a predecessor and successor type of interval j.
A chain $\Gamma = (B_{j_1}, B_{j_2}, \ldots, B_{j_d})$ is consistent if

$$\text{suc}(j) = \text{pre}(j+1), \quad (j = 1, \ldots, n-1).$$

Consistent Chaining Problem

Definition

Let $\mathcal{B} = \{B_1, B_2, \ldots, B_n\}$ and $s_1, \ldots, s_n$ be as above.
In addition, let T be a finite set of types.
For every interval B_j let $\text{pre}(j), \text{suc}(j) \in T$ be a predecessor and successor type of interval j.
A chain $\Gamma = (B_{j_1}, B_{j_2}, \ldots, B_{j_d})$ is consistent if

$$\text{suc}(j) = \text{pre}(j+1), \quad (j = 1, \ldots, n-1).$$

Definition (Consistent Chaining Problem)

For a given set of scored, typed intervals $\mathcal{B}$ find a consistent chain with maximal score.

Consistent Chaining Algorithm

Consistent Chaining Algorithm (without Backtracking)

$P \leftarrow$ sort $\{\ell_1, r_1, \ell_2, r_2, \ldots, \ell_n, r_n\}$ increasingly
$M_t \leftarrow 0$ for all $t \in T$ // initialization
while P not empty **do**
  $b \leftarrow$ remove smallest element in P
  for all j such that $r_j = b$ **do**
    if $S_j > M_{\text{suc}(j)}$ **then**
      $M_{\text{suc}(t)} \leftarrow S_j$
    end if
  end for
  for all j such that $\ell_j = b$ **do**
    $S_j \leftarrow s_j + M_{\text{pre}(j)}$
  end for
end while
output $\max_t M_t$ as score of best chain

Consistent Chaining Algorithm

- algorithm maintains for each t the score M_t of the best chain in which the last interval has successor type t and ends at or before b

Consistent Chaining Algorithm

- algorithm maintains for each t the score M_t of the best chain in which the last interval has successor type t and ends at or before b
- backtracking very similar as in normal chaining algorithm

Consistent Chaining Algorithm

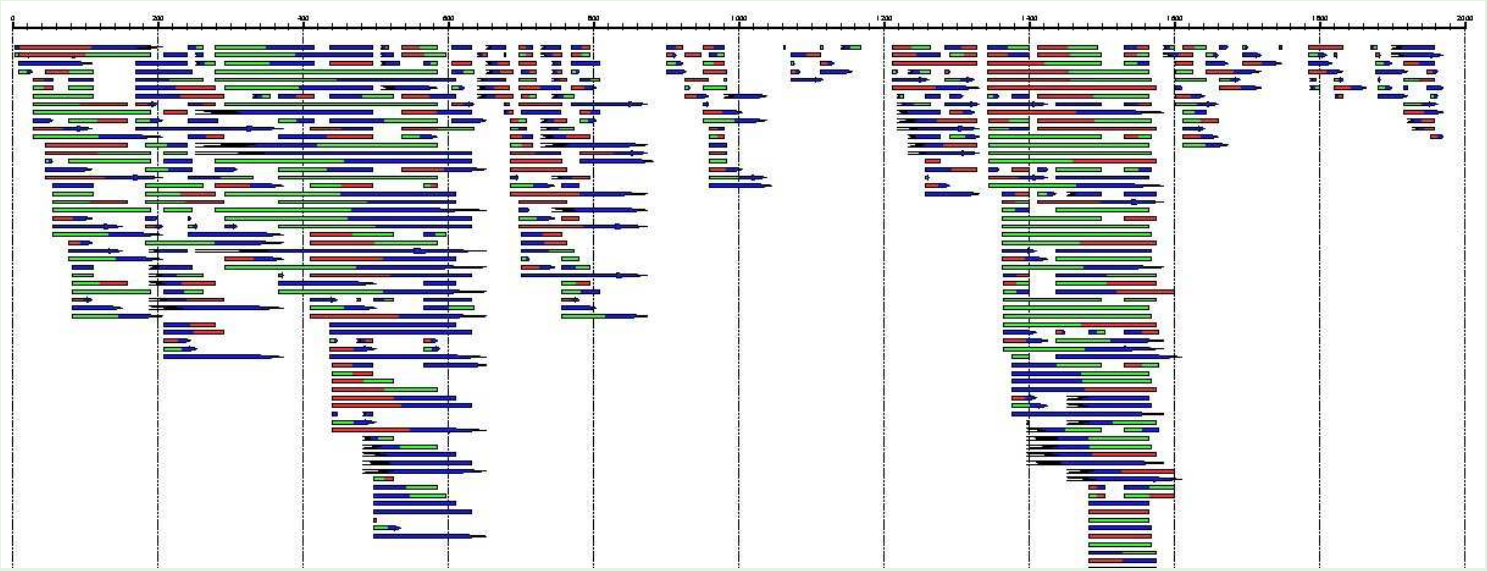
- algorithm maintains for each t the score M_t of the best chain in which the last interval has successor type t and ends at or before b
- backtracking very similar as in normal chaining algorithm
- running time still $O(n \log n)$ if T is considered a constant

Consistent Chaining Algorithm

- algorithm maintains for each t the score M_t of the best chain in which the last interval has successor type t and ends at or before b
- backtracking very similar as in normal chaining algorithm
- running time still $O(n \log n)$ if T is considered a constant
- best chain can now include intervals with negative score

Exon Chaining/Assembly

Example (exon candidates in a DNA of length 2000)



http://monstre1.imim.es/courses

- color at left and right end (red, green, blue) specify exon phase at left and right end
- arrow tips and heads denote start and stop codons

exon candidates of the program GENEID

Exon Chaining/Assembly

Can use Consistent Chaining Algorithm to assemble exon candidates to genes.

exon candidates = intervals

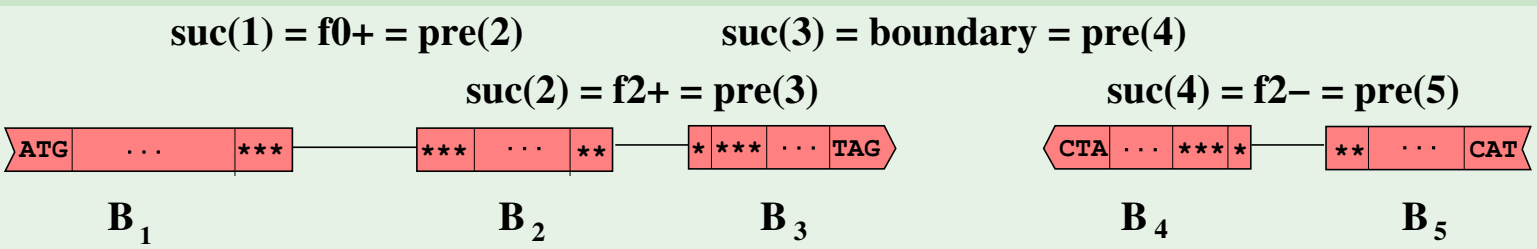
Let T contain the following elements describing a transition type between exons.

boundary	gene boundary
f0+	codon on + strand is split right at boundary
f1+	codon on + strand is split after first base
f2+	codon on + strand is split after second base
f0-	codon on - strand is split right at boundary
f1-	codon on - strand is split after first base
f2-	codon on - strand is split after second base

Define predecessor and successor types of exon candidates so that consistency of chain implies biological consistency of exon sequence.

Consistent Exon Chain

Example



Issues of the Exon Chaining Approach

Problematic:

- introns are not modelled at all:
  - no length distribution considered
  - no difference to intergenic region

Issues of the Exon Chaining Approach

Problematic:

- introns are not modelled at all:
  - no length distribution considered
  - no difference to intergenic region
- UTRs: How can one accomodate for exons like these?

Issues of the Exon Chaining Approach

Problematic:

- introns are not modelled at all:
  - no length distribution considered
  - no difference to intergenic region
- UTRs: How can one accomodate for exons like these?

- dividing by background probability implicitly assumes that there are only two alternatives, e.g. exon $\leftrightarrow$ noncoding but there are more than two alternatives for a region

Reminder: Hidden Markov Model

HMM

- A HMM is a probabilistic model of a word $y = y_1 y_2 \cdots y_n$ ("emission") over some alphabet Σ and of a state sequence $x = (x_1, x_2, \cdots, x_n)$ over some discrete set of states Q.

Reminder: Hidden Markov Model

HMM

- A HMM is a probabilistic model of a word $y = y_1 y_2 \cdots y_n$ ("emission") over some alphabet Σ and of a state sequence $x = (x_1, x_2, \cdots, x_n)$ over some discrete set of states Q.
- The joint distribution of x and y is of the form

$$P(x, y) = \prod_{i=1}^{n} p(x_i | x_{i-1}) \cdot p(y_i | x_i),$$

where the $p(x_i|x_{i-1})$ are the transition probabilities of a Markov chain and the $p(y_i|x_i)$ are alled emission probabilities.

(x_0 is a start state to simplify notation)

Reminder: Hidden Markov Model

Algorithms

- In applications, normally y is observed and x is unobserved/hidden.

Reminder: Hidden Markov Model

Algorithms

- In applications, normally y is observed and x is unobserved/hidden.
- The Viterbi algorithm computes a most likely state sequence $\hat{x} \in \arg\max_x P(x|y)$ in time $O(n)$.

Reminder: Hidden Markov Model

Algorithms

- In applications, normally y is observed and x is unobserved/hidden.
- The Viterbi algorithm computes a most likely state sequence $\hat{x} \in \arg\max_x P(x|y)$ in time $O(n)$.
- The Forward algorithm can be used to compute $P(x, y)$ in time $O(n)$.

Reminder: Hidden Markov Model

Algorithms

- In applications, normally y is observed and x is unobserved/hidden.
- The Viterbi algorithm computes a most likely state sequence $\hat{x} \in \arg\max_x P(x|y)$ in time $O(n)$.
- The Forward algorithm can be used to compute $P(x, y)$ in time $O(n)$.
- The Forward and Backward algorithms can be used to compute posterior probabilities $P(x_i = q|y)$ in time $O(n)$.

Reminder: Generalized Hidden Markov Model

Why GHMMs?

- A HMM is a special case of a GHMM.

Reminder: Generalized Hidden Markov Model

Why GHMMs?

- A HMM is a special case of a GHMM.
- In gene finding and for alignment tasks GHMMs are often used because

Reminder: Generalized Hidden Markov Model

Why GHMMs?

- A HMM is a special case of a GHMM.
- In gene finding and for alignment tasks GHMMs are often used because
  1. they allow a detailed modelling of the length distribution of exons and other biological intervals

Reminder: Generalized Hidden Markov Model

Why GHMMs?

- A HMM is a special case of a GHMM.
- In gene finding and for alignment tasks GHMMs are often used because
  1. they allow a detailed modelling of the length distribution of exons and other biological intervals
  2. they accomodate for “silent” or “delete” states required to model alignment gaps

Definition: Generalized Hidden Markov Model

Definition (Parse)

Let $y = y_1 y_2 \cdots y_n$, Σ, Q be as before.
A parse x of y is a sequence

$$x = ((q_1, v_1), (q_2, v_2), \ldots, (q_t, v_t)),$$

with $q_i \in Q$, $v_i \in \mathbb{N}_0$ such that $v_1 \leq v_2 \leq \cdots \leq v_t = n$.

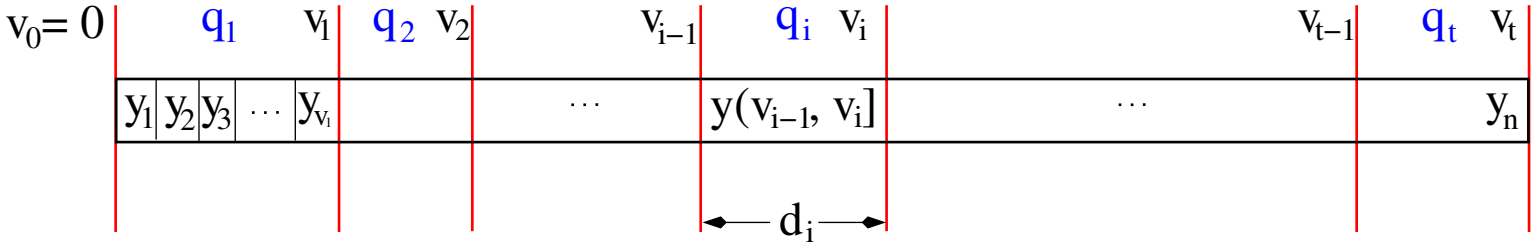
Definition: Generalized Hidden Markov Model

Definition (Parse)

Let $y = y_1 y_2 \cdots y_n$, Σ, Q be as before.
A parse x of y is a sequence

$$x = ((q_1, v_1), (q_2, v_2), \ldots, (q_t, v_t)),$$

with $q_i \in Q$, $v_i \in \mathbb{N}_0$ such that $v_1 \leq v_2 \leq \cdots \leq v_t = n$.



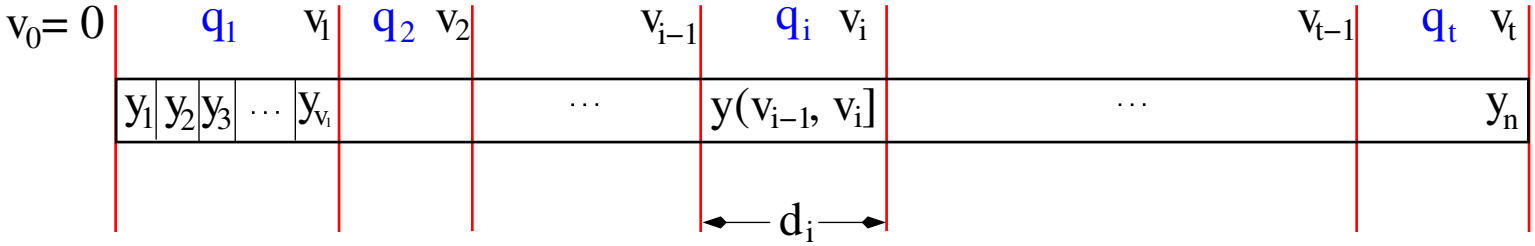
Definition: Generalized Hidden Markov Model

Definition (Parse)

Let $y = y_1 y_2 \cdots y_n$, Σ, Q be as before.
A parse x of y is a sequence

$$x = ((q_1, v_1), (q_2, v_2), \ldots, (q_t, v_t)),$$

with $q_i \in Q$, $v_i \in \mathbb{N}_0$ such that $v_1 \leq v_2 \leq \cdots \leq v_t = n$.



- observe that y decomposes via x into
  $y = y(v_0, v_1]y(v_1, v_2] \cdots y(v_{n-1}, v_n] \qquad (v_0 := 0)$

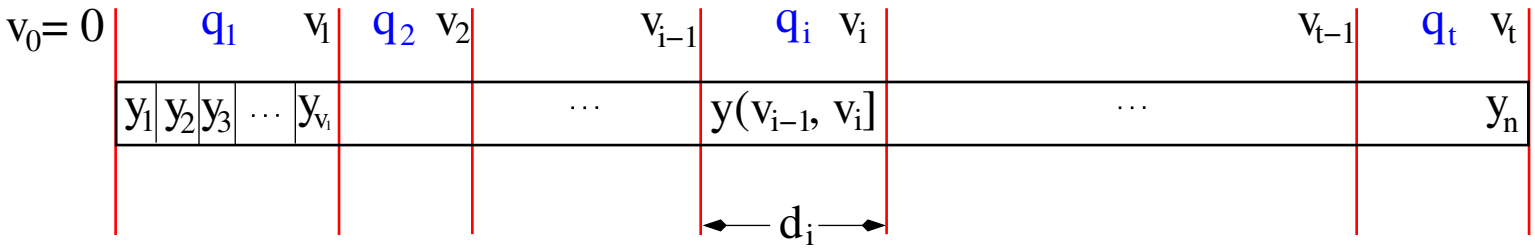
Definition: Generalized Hidden Markov Model

Definition (Parse)

Let $y = y_1 y_2 \cdots y_n$, Σ, Q be as before.
A parse x of y is a sequence

$$x = ((q_1, v_1), (q_2, v_2), \ldots, (q_t, v_t)),$$

with $q_i \in Q$, $v_i \in \mathbb{N}_0$ such that $v_1 \leq v_2 \leq \cdots \leq v_t = n$.



- observe that y decomposes via x into
  $y = y(v_0, v_1]y(v_1, v_2] \cdots y(v_{n-1}, v_n]$ $\qquad (v_0 := 0)$
- we say that state "q_i ends at v_i"

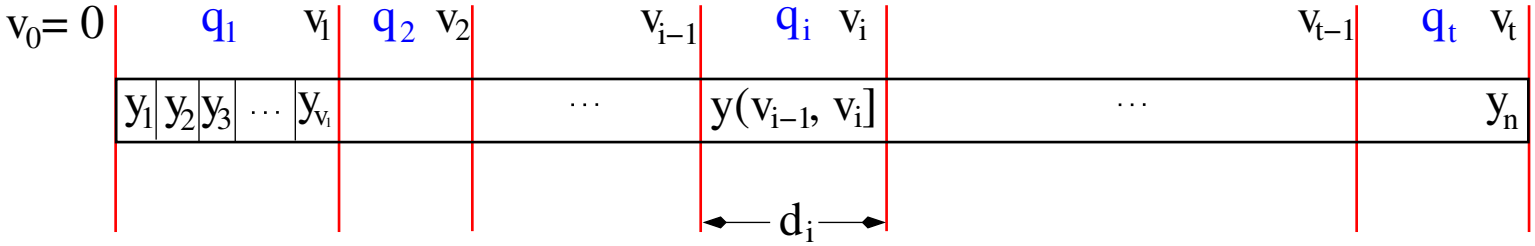
Definition: Generalized Hidden Markov Model

Definition (Parse)

Let $y = y_1 y_2 \cdots y_n$, Σ, Q be as before.
A parse x of y is a sequence

$$x = ((q_1, v_1), (q_2, v_2), \ldots, (q_t, v_t)),$$

with $q_i \in Q$, $v_i \in \mathbb{N}_0$ such that $v_1 \leq v_2 \leq \cdots \leq v_t = n$.



- observe that y decomposes via x into
  $y = y(v_0, v_1]y(v_1, v_2] \cdots y(v_{n-1}, v_n] \qquad (v_0 := 0)$
- we say that state "q_i ends at v_i"
- we call $d_i := v_i - v_{i-1}$ the length of the i-th emission

Definition: Generalized Hidden Markov Model

Definition (GHMM)

A GHMM is a joint distribution of a word y and a parse x of y of the form

$$P(x, y) = \prod_{i=1}^{t} P_{\text{trans}}(q_i | q_{i-i}) \cdot P_{\text{emi}}(y(v_{i-1}, v_i] | q_i),$$

where $P_{\text{trans}}(\cdot|q)$ is a probability distribution (transition probabilities) over Q for all $q \in Q$ and where $P_{\text{emi}}(\cdot|q)$ is a probability distribution (emission probabilities) over Σ^* for all $q \in Q$.

q_0 is a special start state

$\Sigma^* = \{\text{all strings with letters in } \Sigma\}$ (includes empty string)

Remark: We explicitly allow $d_i = 0$. A state q with $P_{\text{emi}}(\epsilon|q) = 1$ is called a silent state (ϵ is the empty string of length 0).

Delineation of HMM

When is a GHMM called a HMM?

- A HMM is a GHMM in which $d_i \equiv 1$ for all i, i.e. all emissions are a single character. In that special case the parse x can be identified with the state sequence, which has the same length as y
- Sometimes in the literature a GHMM, in which $d_i \in \{0, 1\}$, is still called a HMM only with some special modifications to the algorithms. Example: "delete" state in profile HMMs

Algorithms for GHMM

Algorithms

1. Usually, the word y is observed.
   Now: A concatenation of the emissions, not the sequence of emissions.
   Contrast to HMM: The emissions cannot be inferred from y alone.

Algorithms for GHMM

Algorithms

1. Usually, the word y is observed.
   Now: A concatenation of the emissions, not the sequence of emissions.
   Contrast to HMM: The emissions cannot be inferred from y alone.
2. x is unobserved, neither the states nor their boundaries are known.

Algorithms for GHMM

Algorithms

1. Usually, the word y is observed.
   Now: A concatenation of the emissions, not the sequence of emissions.
   Contrast to HMM: The emissions cannot be inferred from y alone.
2. x is unobserved, neither the states nor their boundaries are known.
3. Analogous Viterbi, Forward and Backward algorithms exists that all run in $O(n^2)$. Important special case: they run in $O(n)$ if all d_i are bounded from above by a constant.

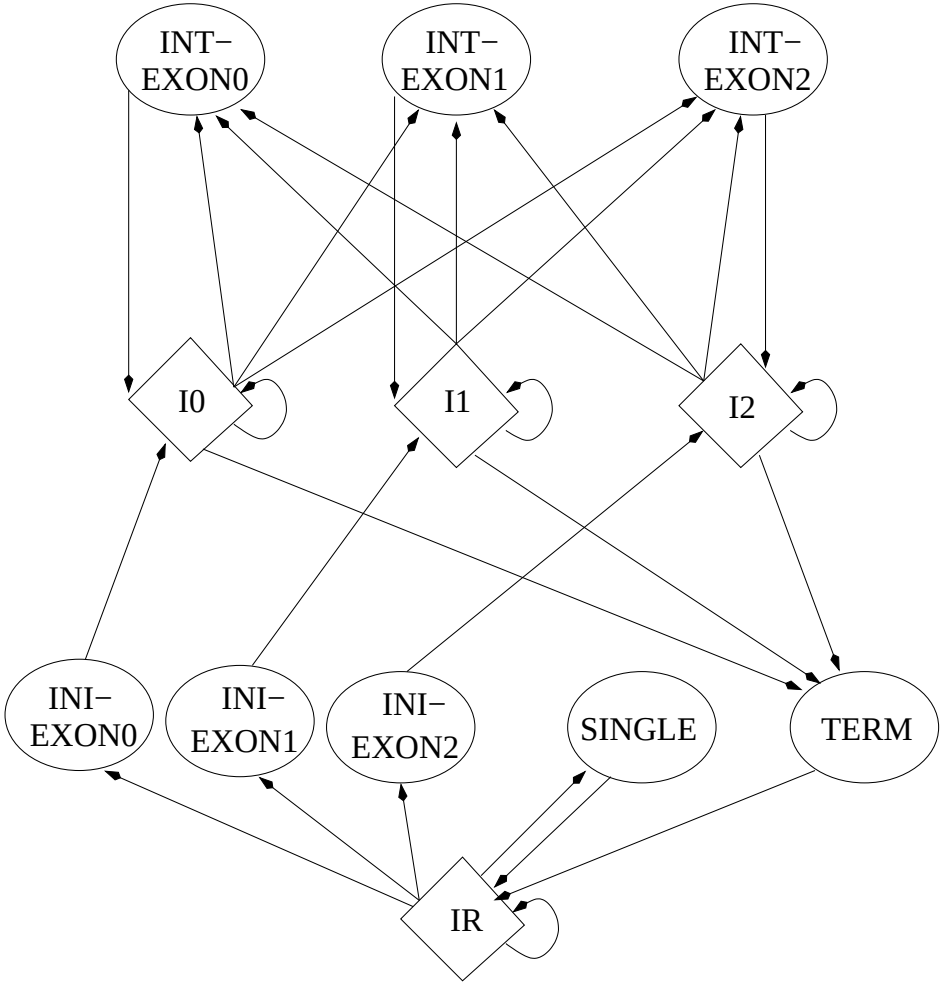
Algorithms for GHMM

Algorithms

1. Usually, the word y is observed.
   Now: A concatenation of the emissions, not the sequence of emissions.
   Contrast to HMM: The emissions cannot be inferred from y alone.
2. x is unobserved, neither the states nor their boundaries are known.
3. Analogous Viterbi, Forward and Backward algorithms exists that all run in $O(n^2)$. Important special case: they run in $O(n)$ if all d_i are bounded from above by a constant.
4. A prerequisite for points 3 above is that no loops of states with just empty-word-emissions are possible.
   We will ensure that by the design of the model topology.

A Simple GHMM for Gene Finding: Model Topology

Model for (multiple) eukaryotic genes on forward strand:



(Arrows denote the transitions with non-zero transition probability.)

What (Most) Eukaryotic Species Have in Common?

In Common:

- same genetic code, including start and stop codons

What (Most) Eukaryotic Species Have in Common?

In Common:

- same genetic code, including start and stop codons
- genes can have introns, may have many

What (Most) Eukaryotic Species Have in Common?

In Common:

- same genetic code, including start and stop codons
- genes can have introns, may have many
- genes rarely overlap in sequence

What (Most) Eukaryotic Species Have in Common?

In Common:

- same genetic code, including start and stop codons
- genes can have introns, may have many
- genes rarely overlap in sequence
- introns start almost always with GT, end with AG (some introns GC/AG)

What (Most) Eukaryotic Species Have in Common?

In Common:

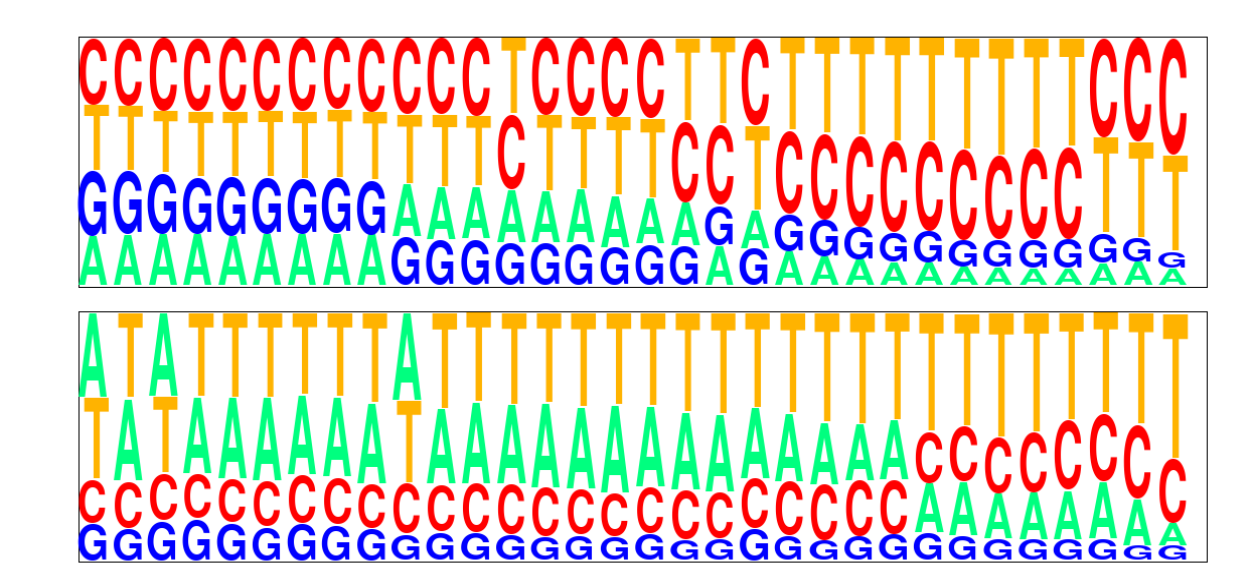
- same genetic code, including start and stop codons
- genes can have introns, may have many
- genes rarely overlap in sequence
- introns start almost always with GT, end with AG (some introns GC/AG)
- more non-coding sequence than coding sequence

How Species-Specific Must Gene Finding Models Be?

Differences:

- distribution at signals, e.g. branch point region

top: human / bottom: fly



How Species-Specific Must Gene Finding Models Be?

Differences:

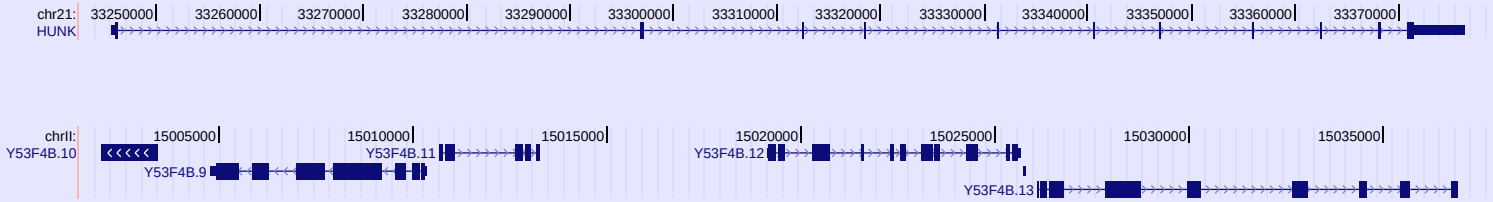
- distribution at signals, e.g. branch point region
- GC content highly variable

How Species-Specific Must Gene Finding Models Be?

Differences:

- distribution at signals, e.g. branch point region
- GC content highly variable
- number and length distribution of introns

top: human / bottom: *C. elegans*



How Species-Specific Must Gene Finding Models Be?

Differences:

- distribution at signals, e.g. branch point region
- GC content highly variable
- number and length distribution of introns
- length distribution of UTRs

How Species-Specific Must Gene Finding Models Be?

Differences:

- distribution at signals, e.g. branch point region
- GC content highly variable
- number and length distribution of introns
- length distribution of UTRs
- gene density

Training: Estimate Species-Specific Parameters

"Training Set"

- input: set of annotated sequences

$$(x^{(k)}, y^{(k)})_{k=1,\ldots,N},$$

  such that the parse $x^{(k)}$ represents the gene structure of DNA sequence $y^{(k)}$.
- frequently a few hundred genes constructed from cDNA alignments

Pair HMM versus standard HMM

Pair HMM

- same concept of hidden states

Pair HMM versus standard HMM

Pair HMM

- same concept of hidden states
- two observed sequences y and z instead of just one

Pair HMM versus standard HMM

Pair HMM

- same concept of hidden states
- two observed sequences y and z instead of just one
- an association between character pairs y_i and z_j is usually sought but a priori not known

Pair HMM versus standard HMM

Pair HMM

- same concept of hidden states
- two observed sequences y and z instead of just one
- an association between character pairs y_i and z_j is usually sought but a priori not known
- typical Bioinformatics applications: alignments, comparative gene finding

Biparse

Definition (Biparse)

Let Q be a finite set (of states).
Let $y = y_1 y_2 \cdots y_n$ and $z = z_1 z_2 \cdots z_m$ be two sequences over an alphabet Σ of lengths n and m, respectively.
A biparse x of y and z is a sequence

$$x = ((q_1, v_1, w_1), (q_2, v_2, w_2), \ldots, (q_t, v_t, w_t)),$$

with $q_i \in Q$, $v_i, w_i \in \mathbb{N}_0$ such that
$v_1 \leq v_2 \leq \cdots \leq v_t = n$ and $w_1 \leq w_2 \leq \cdots \leq w_t = m$.

Biparse

Definition (Biparse)

Let Q be a finite set (of states).
Let $y = y_1 y_2 \cdots y_n$ and $z = z_1 z_2 \cdots z_m$ be two sequences over an alphabet Σ of lengths n and m, respectively.
A biparse x of y and z is a sequence

$$x = ((q_1, v_1, w_1), (q_2, v_2, w_2), \ldots, (q_t, v_t, w_t)),$$

with $q_i \in Q$, $v_i, w_i \in \mathbb{N}_0$ such that
$v_1 \leq v_2 \leq \cdots \leq v_t = n$ and $w_1 \leq w_2 \leq \cdots \leq w_t = m$.

- a biparse segments 2 sequences into the same number of segments
- each segment pair $y(v_{i-1}, v_i], z(w_{i-1}, w_i]$ corresponds a single state q_i

Definition: Pair HMM

Definition (Pair HMM)

A Pair HMM is a joint distribution of two words y and z and a biparse x of them of the form

$$P(x, y, z) = \prod_{i=1}^{t} P_{\text{trans}}(q_i | q_{i-i}) \cdot P_{\text{emi}}(y(v_{i-1}, v_i], z(w_{i-1}, w_i] | q_i),$$

where $P_{\text{trans}}(\cdot|q)$ is a probability distribution (transition probs) over Q for all $q \in Q$ and where
$P_{\text{emi}}(\cdot|q)$ is a probability distr. (emission probs) over $\Sigma^* \times \Sigma^*$ for all $q \in Q$.

$q_0 \in Q$ is a special start state

- Analogous to GHMM, just 2 "simultaneous" emissions instead of 1.
- In practice, P_{emi} often is symmetric:
  $P_{\text{emi}}(a, b|q) = P_{\text{emi}}(b, a|q)$ (fewer parameters to train)

Viterbi Algorithm for Pair HMMs

Definition (Viterbi Variables)

For $q \in Q, 0 \leq \ell \leq n, 0 \leq r \leq m$ define the Viterbi variable

$$\gamma_{q,\ell,r} := \max_{\substack{x \text{ biparse} \\ \text{that ends in} \\ (q,\ell,r)}} P(x, y(0,\ell], z(0,r]).$$

Viterbi Algorithm for Pair HMMs

Definition (Viterbi Variables)

For $q \in Q, 0 \leq \ell \leq n, 0 \leq r \leq m$ define the Viterbi variable

$$\gamma_{q,\ell,r} := \max_{\substack{x \text{ biparse} \\ \text{that ends in} \\ (q,\ell,r)}} P(x, y(0,\ell], z(0,r]).$$

Interpretation

$\gamma_{q,\ell,r}$ is the probability of the most likely parse of y up to ℓ and of z up to r that ends in state q.

Viterbi Recursion

Viterbi Recursion

$$\gamma_{q,\ell,r} = \max_{\substack{q' \in Q \\ 0 \le \ell' \le \ell \\ 0 \le r' \le r}} \gamma_{q',\ell',r'} P_{\text{trans}}(q|q') P_{\text{emi}}(y(\ell',\ell], z(r',r]|q)$$

Here, for convenience we define

$$\gamma_{q_0,0,0} = 1, \qquad \gamma_{q,0,0} = 0 \quad \forall q \neq q_0.$$

Viterbi Recursion

Viterbi Recursion

$$\gamma_{q,\ell,r} = \max_{\substack{q' \in Q \\ 0 \le \ell' \le \ell \\ 0 \le r' \le r}} \gamma_{q',\ell',r'} P_{\text{trans}}(q|q') P_{\text{emi}}(y(\ell',\ell], z(r',r]|q)$$

Here, for convenience we define

$$\gamma_{q_0,0,0} = 1, \qquad \gamma_{q,0,0} = 0 \quad \forall q \neq q_0.$$

Assumption

Never the empty string is emitted simultaneously in both sequences:

$$P_{\text{emi}}(\epsilon, \epsilon|q) = 0 \quad \forall q \in Q$$

Viterbi Recursion

Viterbi Recursion

$$\gamma_{q,\ell,r} = \max_{\substack{q' \in Q \\ 0 \le \ell' \le \ell \\ 0 \le r' \le r}} \gamma_{q',\ell',r'} P_{\text{trans}}(q|q') P_{\text{emi}}(y(\ell',\ell], z(r',r]|q)$$

Here, for convenience we define

$$\gamma_{q_0,0,0} = 1, \qquad \gamma_{q,0,0} = 0 \quad \forall q \neq q_0.$$

Assumption

Never the empty string is emitted simultaneously in both sequences:

$$P_{\text{emi}}(\epsilon, \epsilon|q) = 0 \quad \forall q \in Q$$

- is anyway the case in our applications
- is sufficient condition that the Viterbi recursion can be iteratively computed

Viterbi Algorithm for Pair HMMs

```
initialize γ_{q_0,0,0} ← 1, γ_{q,0,0} ← 0   ∀q ∈ Q \ {q_0}
for ℓ = 0 to n do
    for r = 0 to m do
        for all q ∈ Q do
            if ℓ ≠ 0 or r ≠ 0 then
                update γ_{q,ℓ,r} according to Viterbi recursion
                pre(q, ℓ, r) ← (q′, ℓ′, r′) // arg max from Viterbi recursion
            end if
        end for
    end for
end for
// backtracking starts
x ← ()
q ← arg max_{q′∈Q} γ_{q′,n,m},   ℓ ← n, r ← m
while ℓ > 0 or r > 0 do
    add (q, ℓ, r) at front of x
    (q, ℓ, r) = pre(q, ℓ, r)
end while
output x as a best biparse of y and z
```

Running Time

- in general:

Running Time

- in general: $O(n^2 m^2)$
- if emissions are bounded by d:
  $P_{emi}(w, w'|q) = 0, \quad \forall w, w' \in \Sigma^* : |w| > d \text{ or } |w'| > d, \forall q \in Q$
  we can shortcut recursion:

$$\gamma_{q,\ell,r} = \max_{\substack{q' \in Q \\ \max\{0, \ell - d\} \leq \ell' \leq \ell \\ \max\{0, \ell - d\} \leq r' \leq r}} \gamma_{q',\ell',r'} P_{\text{trans}}(q|q') P_{\text{emi}}(y(\ell', \ell], z(\ell', \ell]|q)$$

then running time is

Running Time

- in general: $O(n^2 m^2)$
- if emissions are bounded by d:
  $P_{emi}(w, w'|q) = 0, \quad \forall w, w' \in \Sigma^* : |w| > d \text{ or } |w'| > d, \forall q \in Q$
  we can shortcut recursion:

$$\gamma_{q,\ell,r} = \max_{\substack{q' \in Q \\ \max\{0, \ell - d\} \leq \ell' \leq \ell \\ \max\{0, \ell - d\} \leq r' \leq r}} \gamma_{q',\ell',r'} P_{\text{trans}}(q|q') P_{\text{emi}}(y(\ell', \ell], z(\ell', \ell]|q)$$

then running time is $O(d^2 nm)$

Running Time

- in general: $O(n^2 m^2)$
- if emissions are bounded by d:
  $P_{emi}(w, w'|q) = 0, \quad \forall w, w' \in \Sigma^* : |w| > d \text{ or } |w'| > d, \forall q \in Q$
  we can shortcut recursion:

$$\gamma_{q,\ell,r} = \max_{\substack{q' \in Q \\ \max\{0, \ell - d\} \leq \ell' \leq \ell \\ \max\{0, \ell - d\} \leq r' \leq r}} \gamma_{q',\ell',r'} P_{\text{trans}}(q|q') P_{\text{emi}}(y(\ell', \ell], z(\ell', \ell]|q)$$

  then running time is $O(d^2 nm)$
- very important special case $d = 1$: running time = $O(nm)$

Running Time

- in general: $O(n^2 m^2)$
- if emissions are bounded by d:
  $P_{emi}(w, w'|q) = 0, \quad \forall w, w' \in \Sigma^* : |w| > d \text{ or } |w'| > d, \forall q \in Q$
  we can shortcut recursion:

$$\gamma_{q,\ell,r} = \max_{\substack{q' \in Q \\ \max\{0, \ell - d\} \leq \ell' \leq \ell \\ \max\{0, \ell - d\} \leq r' \leq r}} \gamma_{q',\ell',r'} P_{\text{trans}}(q|q') P_{\text{emi}}(y(\ell', \ell], z(\ell', \ell]|q)$$

  then running time is $O(d^2 nm)$
- very important special case $d = 1$: running time = $O(nm)$
- further heuristics to reduce running time possible:
  compute Viterbi recursion only for subset of $(\ell, r) \in (0, n] \times (0, m]$,
  assume it vanishes elsewhere

Conservation of Gene Structure and Sequence

Observation

Protein sequences and rough structure of genes are often conserved between species that are tens of millions of years separated.

Example (Human-Mouse: 75 million years)

- 95% of orthologous gene pairs have same number of exons in human and mouse

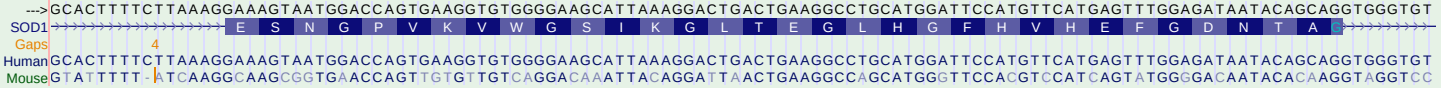
Conservation of Gene Structure and Sequence

Observation

Protein sequences and rough structure of genes are often conserved between species that are tens of millions of years separated.

Example (Human-Mouse: 75 million years)

- 95% of orthologous gene pairs have same number of exons in human and mouse
- coding sequence to $\approx$ 85% identical



Conservation of Gene Structure and Sequence

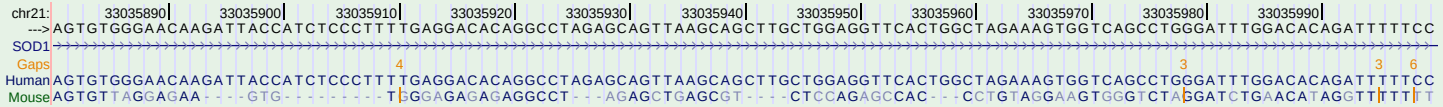
Observation

Protein sequences and rough structure of genes are often conserved between species that are tens of millions of years separated.

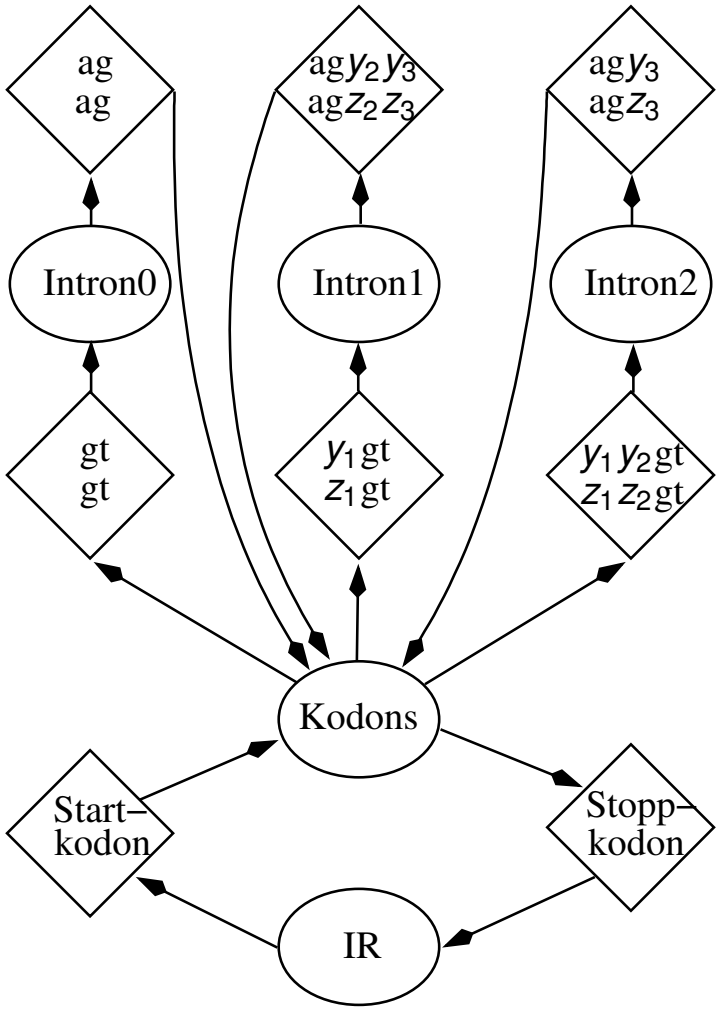
Example (Human-Mouse: 75 million years)

- 95% of orthologous gene pairs have same number of exons in human and mouse
- coding sequence to $\approx$ 85% identical

- noncoding sequence to $\approx$ 35% identical



A Simple Pair HMM for Eukaryotic Gene Finding



- assume 1-to-1 correspondence between exons
- all states emit 2 sequences
- ◇ -shaped states emit fixed-length and equal-length seqs
- splice site and “Kodon” states accomodate for conservation between the two species