Gene Annotation

Michael Schatz

March 1, 2018

Lecture 10: Applied Comparative Genomics

Assignment 4: Due Thursday March 1

Assignment 4: Read mapping and variant calling

Assignment Date: Thursday, Feb. 22, 2018
Due Date: Thursday, Mar. 1, 2018 @ 11:59pm

Assignment Overview

In this assignment, you will align reads to a reference genome to call SNPs and short indels. Then, you will perform an experiment to empirically determine the "mappability" of a genomic region. Finally, you will investigate some empirical behavior of the binomial test for heterozygous variant calling.
As a reminder, any questions about the assignment should be posted to Piazza. Don't forget to read the **Resources** section at the bottom of the page!

Question 1. Small Variant Analysis [XX pts]

Download chromosome 22 from build 38 of the human genome from here:
http://hgdownload.cse.ucsc.edu/goldenPath/hg38/chromosomes/chr22.fa.gz

Download the read set from here:
http://schatzlab.cshl.edu/data/teaching/sample.tgz

For this question, you may find this tutorial helpful:
http://clavius.bc.edu/~erik/CSHL-advanced-sequencing/freebayes-tutorial.html

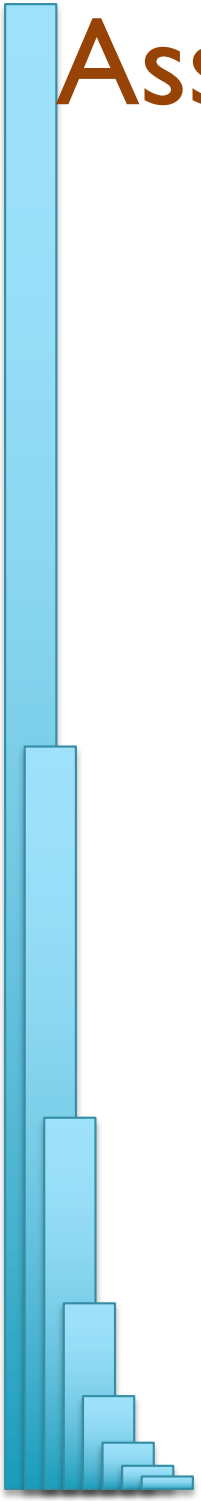
- 1a. How many reads align to the reference? How many reads did not align? How many aligned reads had a mate that did not align (AKA singletons)? Count each read in a pair separately.
  [Hint: Build the index using `bowtie2-build`, align reads using `bowtie2`, analyze with `samtools flagstat`.]
- 1b. How many reads are mapped to the reverse strand? Count each read in a pair separately.
  [Hint: Find out what SAM flags mean here and use samtools view.]
- 1c. How many high-quality (QUAL > 20) single nucleotide and indel variants does the sample have? Of the high-quality SNPs, what is the transition / transversion ratio? Of the indels, how many are insertions and how many are deletions?
  [Hint: Identify variants using `freebayes` - sort the SAM file first. Filter using `bcftools filter`, and summarize using `bcftools stats`.]
- 1d. Does the sample have any nonsense or missense mutations?
  [Hint: try the Variant Effect Predictor using the `Gencode basic transcripts`]

Question 2. Read Mapping Uncertainty [XX pts]

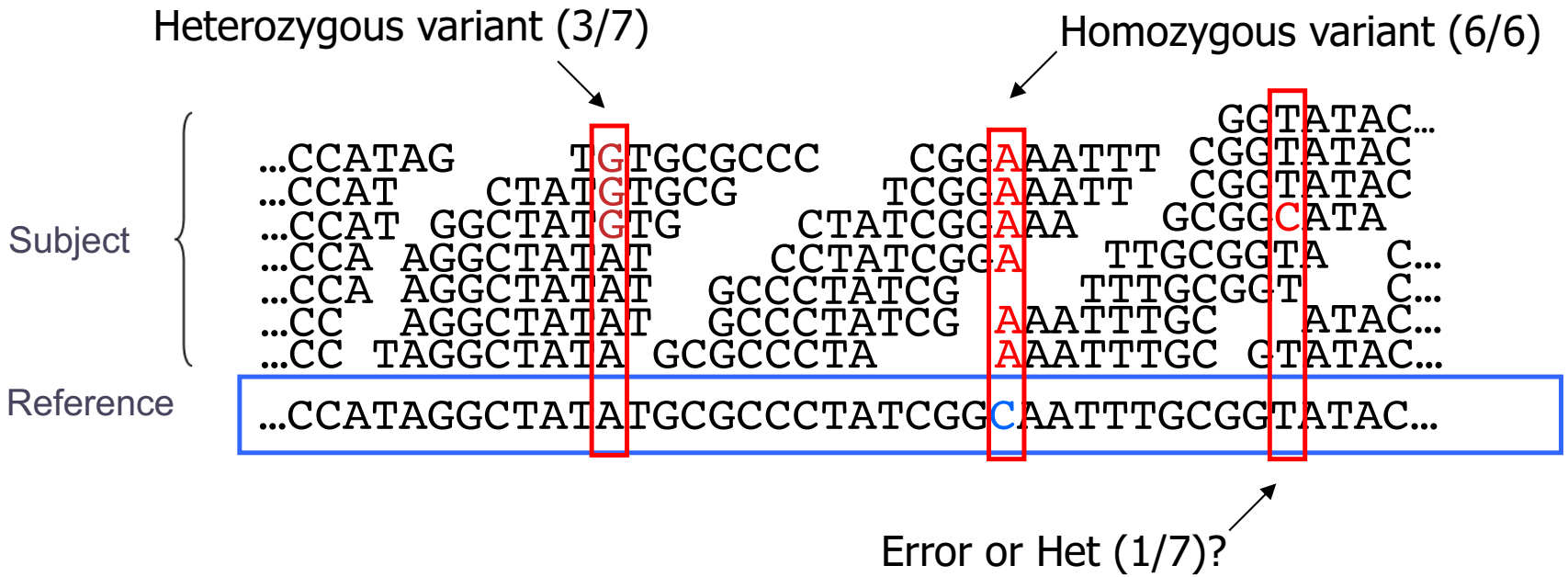
For the region chr22:21000000-22000000 of the reference sequence for chromosome 22, extract every substring of length 35. Format the substrings as a FASTA file and use read names that indicate the origin. (No need to construct quality values or read pairs: use bowtie2 with `-f` and `-U` respectively). Make a new index and align these "reads" to chr22:21000000-22000000.
[Hint: On the command line or in a script, load the sequence once and extract substrings in a loop.]

- 2a. How many reads align more than one time to the reference? How many reads did not align?

Assignment 5: Due Thursday March 8



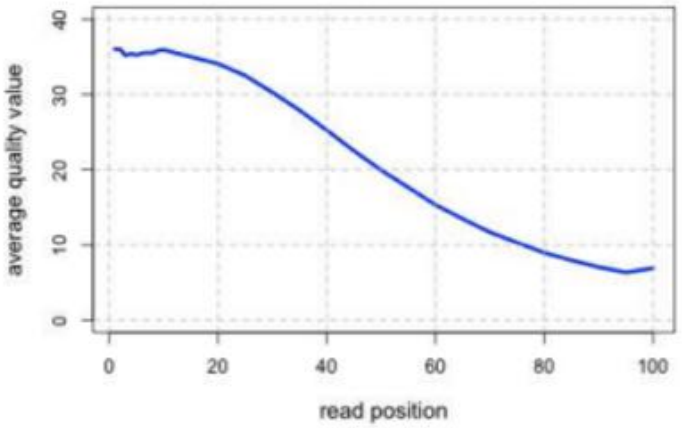
Genotyping Theory



- If there were no sequencing errors, identifying SNPs would be very easy: any time a read disagrees with the reference, it must be a variant!

- Sequencing instruments make mistakes
  - Quality of read decreases over the read length

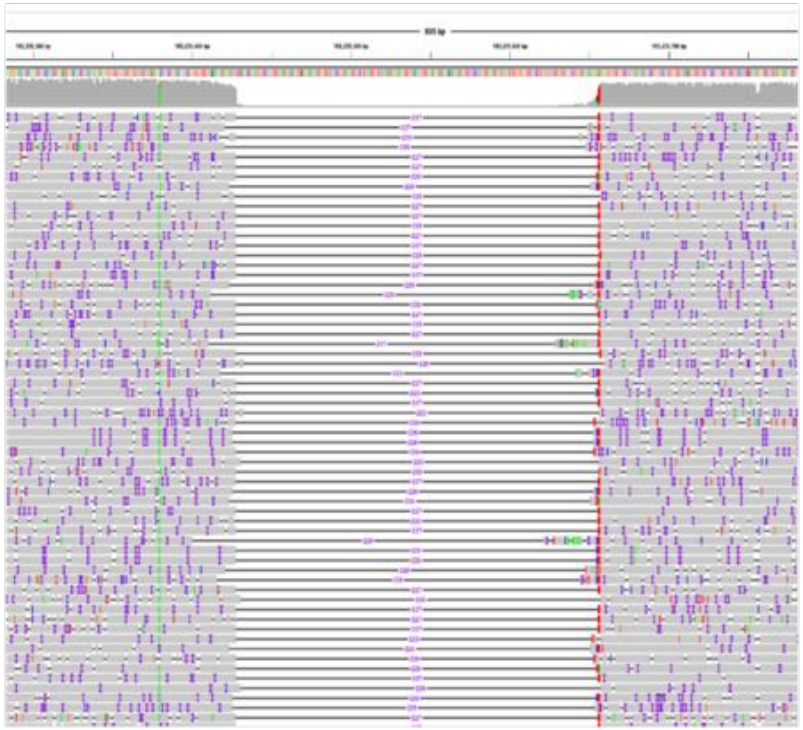
- A single read differing from the reference is probably just an error, but it becomes more likely to be real as we see it multiple times



NGMLR + Sniffles

BWA-MEM:

NGMLR:

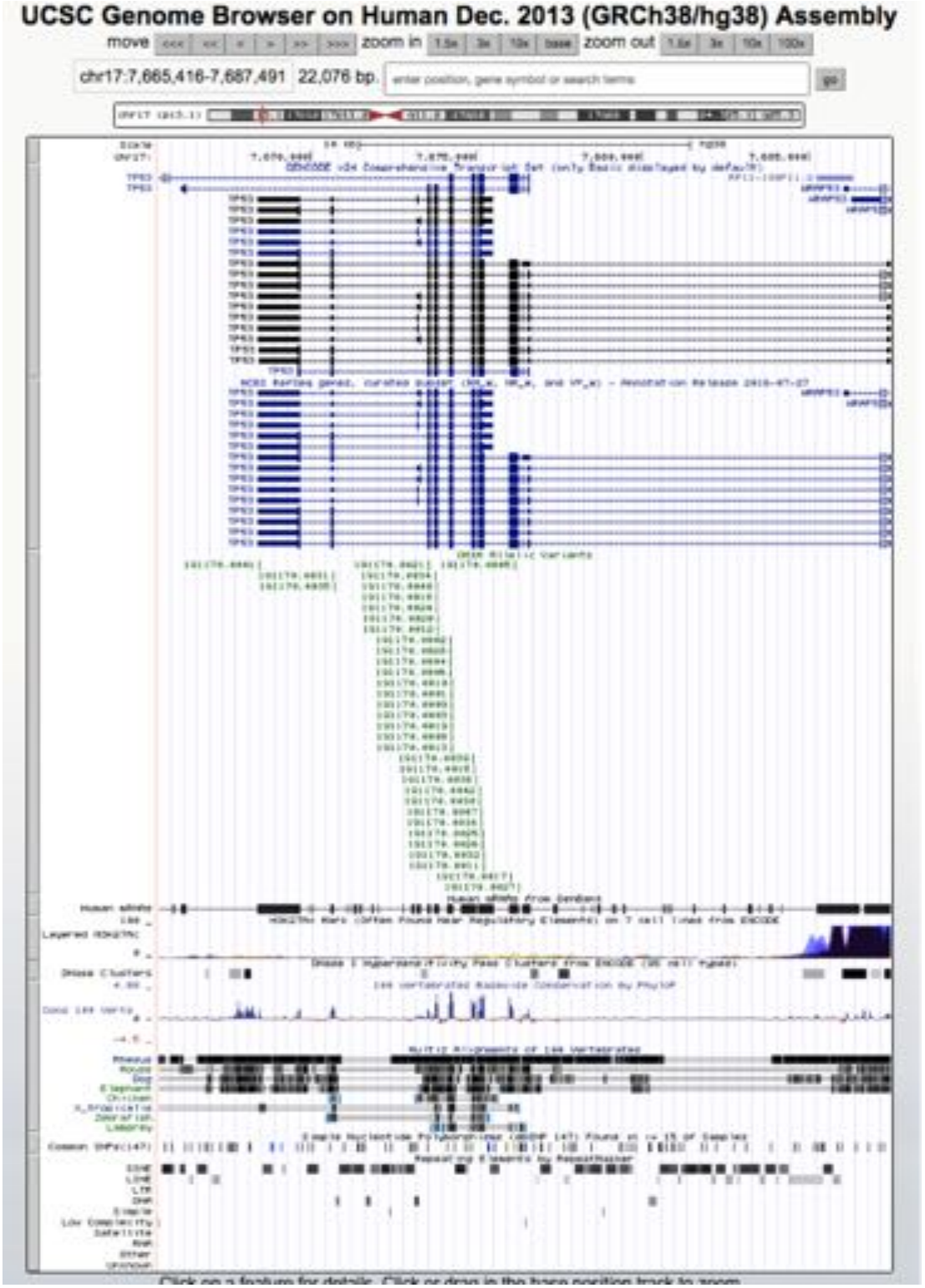


NGMLR: Convex gap penalty to balance frequent small sequencing errors with larger SVs
Sniffles: Scan within and between split reads to accurately find SVs (Ins, Del, Dup, Inv, Trans)
Mendelian concordance >95%, experimental validation also very high

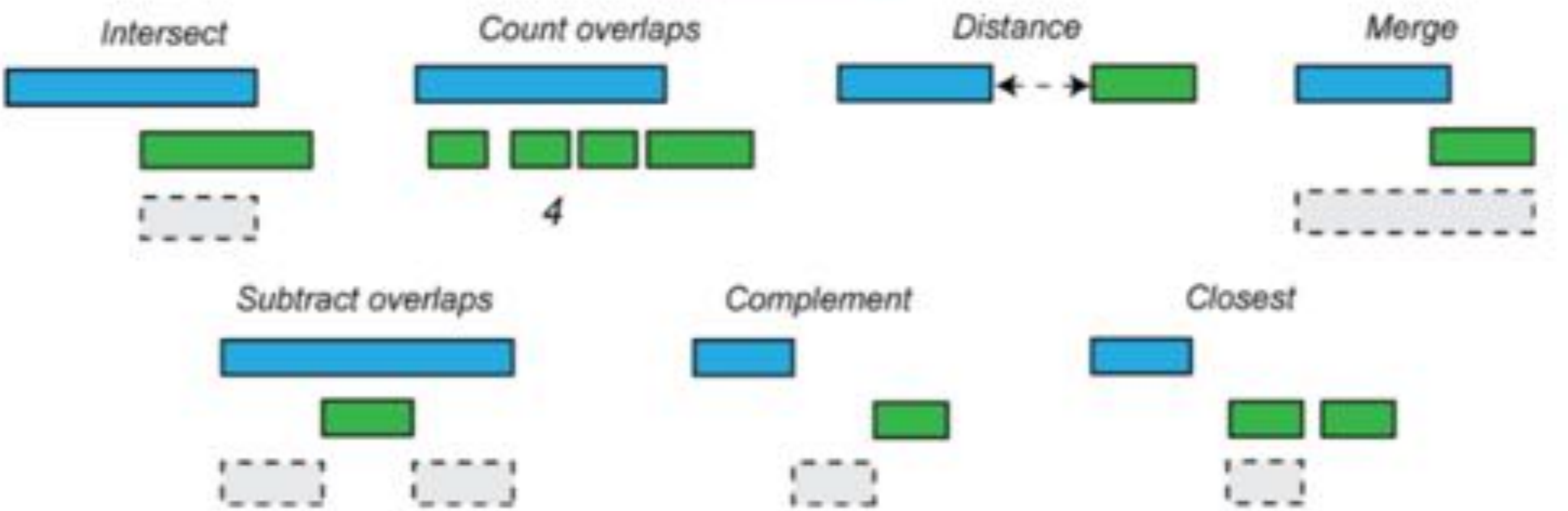
Accurate detection of complex structural variations using single molecule sequencing
Sedlazeck, Rescheneder et al (2018) *Nature Methods. In Press*

What are genome intervals?

- Genetic variation:
  - SNPs: 1bp
  - Indels: 1-50bp
  - SVs: >50bp
- Genes:
  - exons, introns, UTRs, promoters
- Conservation
- Transposons
- Origins of replication
- TF binding sites
- CpG islands
- Segmental duplications
- Sequence alignments
- Chromatin annotations
- Gene expression data
- …
- ***Your own observations and data: put them into context!***



BEDTools to the rescue!

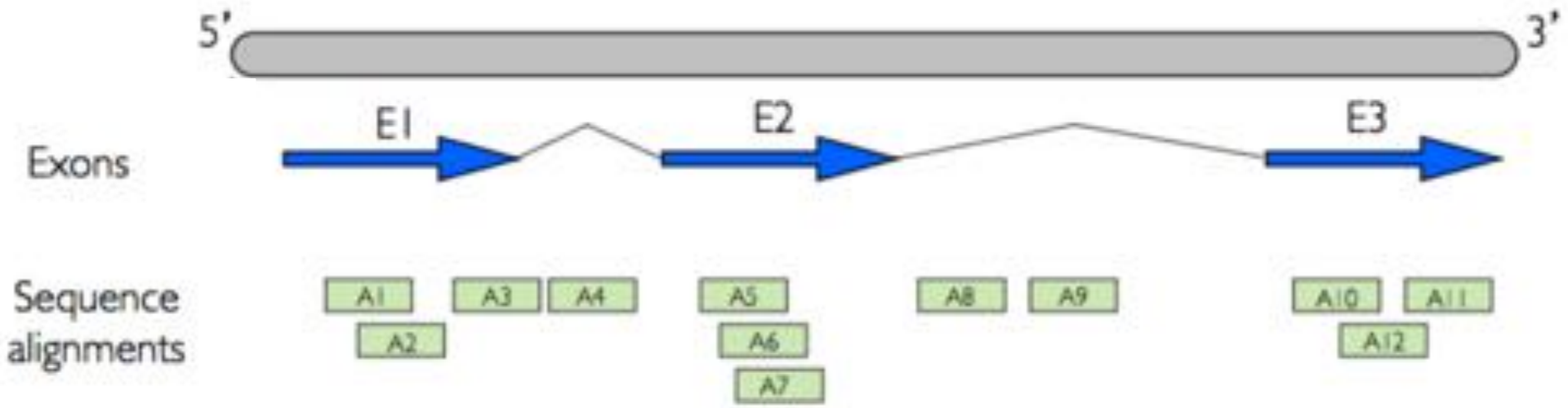


BEDTools commands

```
annotate
bamtobed
bamtofastq
bed12tobed6
bedpetobam
bedtobam
closest
cluster
complement
coverage
expand
flank
fisher
genomecov
getfasta
groupby
groupby
igv
intersect
jaccard
links
makewindows
map
maskfasta
merge
multicov
multiinter
nuc
overlap
pairtobed
pairtopair
random
reldist
shift
shuffle
slop
sort
subtract
tag
unionbedg
window
```

http://bedtools.readthedocs.io/en/latest/content/bedtools-suite.html

BEDTools Performance



How many reads are aligned to exonic sequences?

```
$ awk '{if ($3=="exon"){print}}' gencode.v21.annotation.gff3 | wc -l
1162114
```

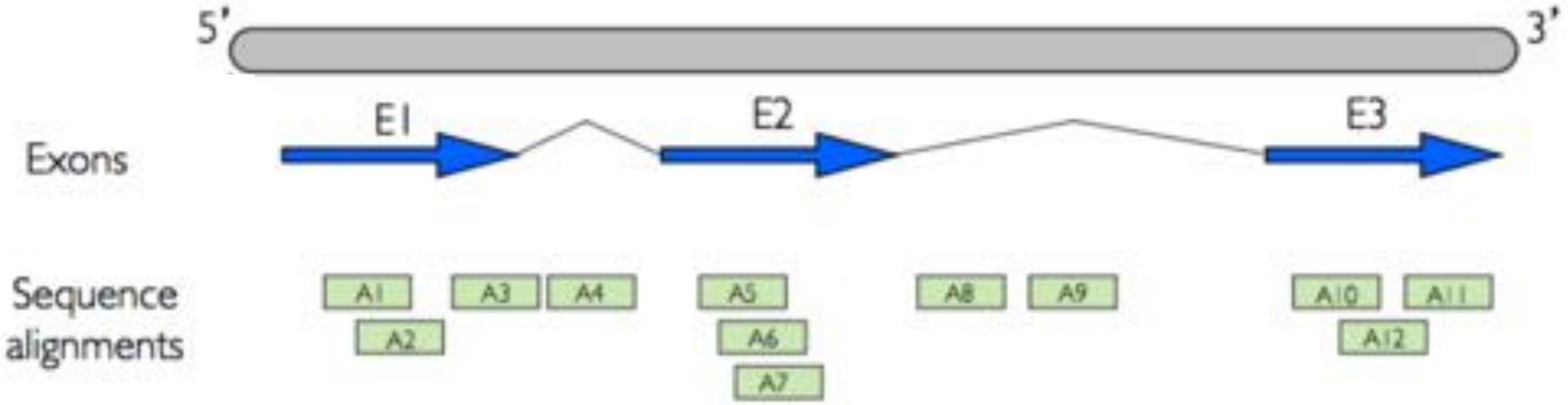
```
if ((read.start <= exon.end) && (read.end >= exon.start)) { print “in exon!”; }
```

How many comparison would a brute force approach take to scan a 30x dataset?

30x3Gb = 90Gbp / 100bp reads = 900M reads

900M reads x 1.1M exons = 990MM comparisons! ☹

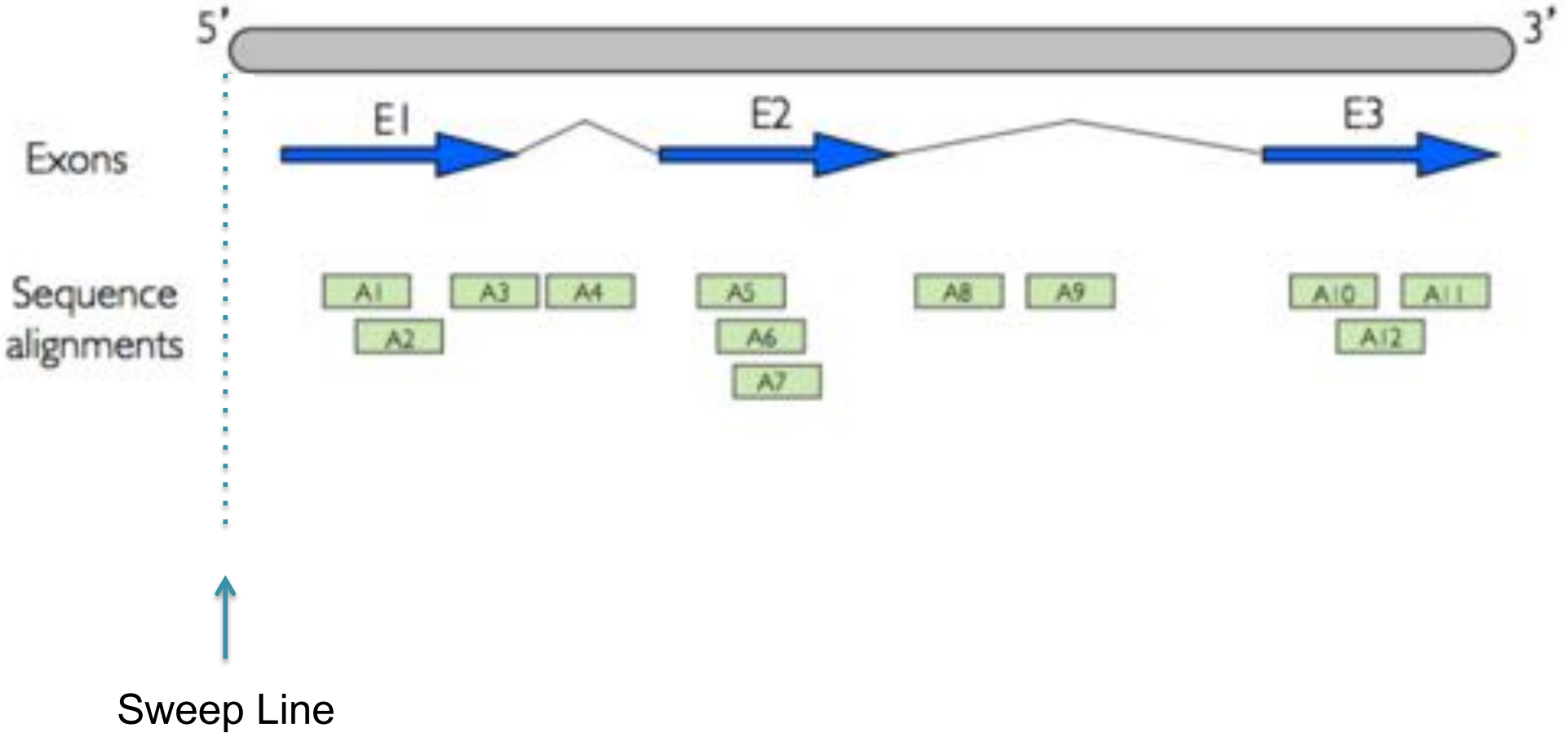
BEDTools Performance



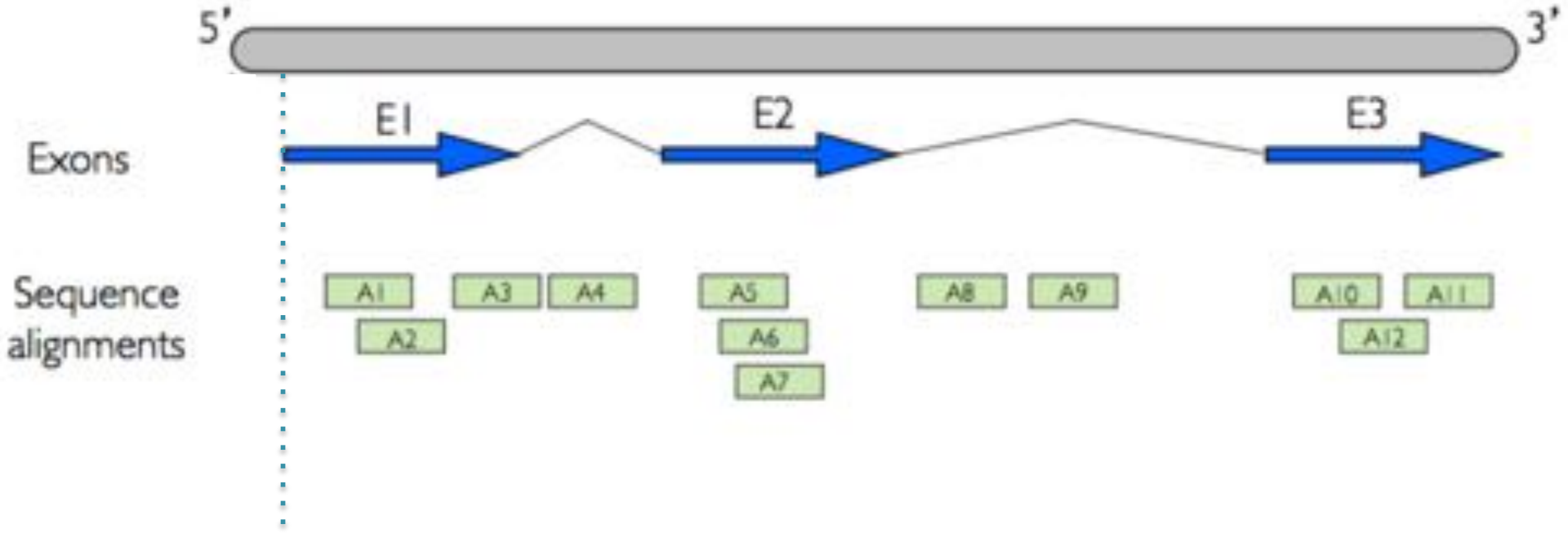
Assume datasets are sorted by chromosome and start position
`samtools sort` to sort alignments, unix `sort` to sort BED file

Any ideas?

Plane Sweep to the Rescue!



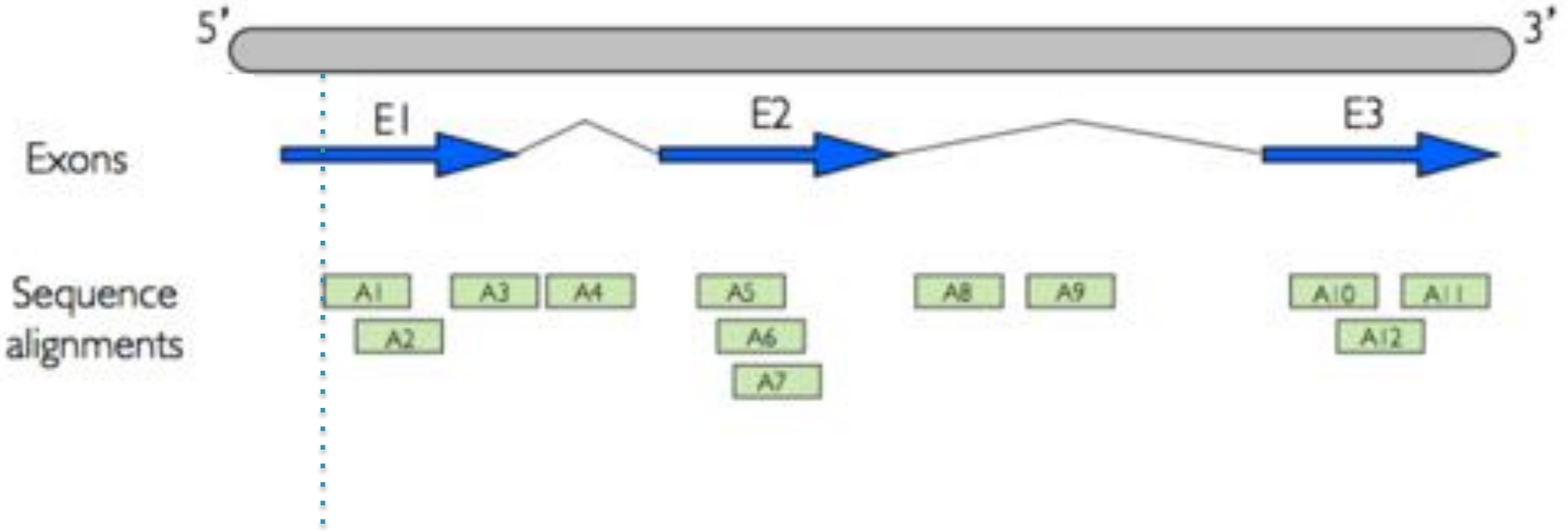
Plane Sweep to the Rescue!



Start of E1
E1 is active

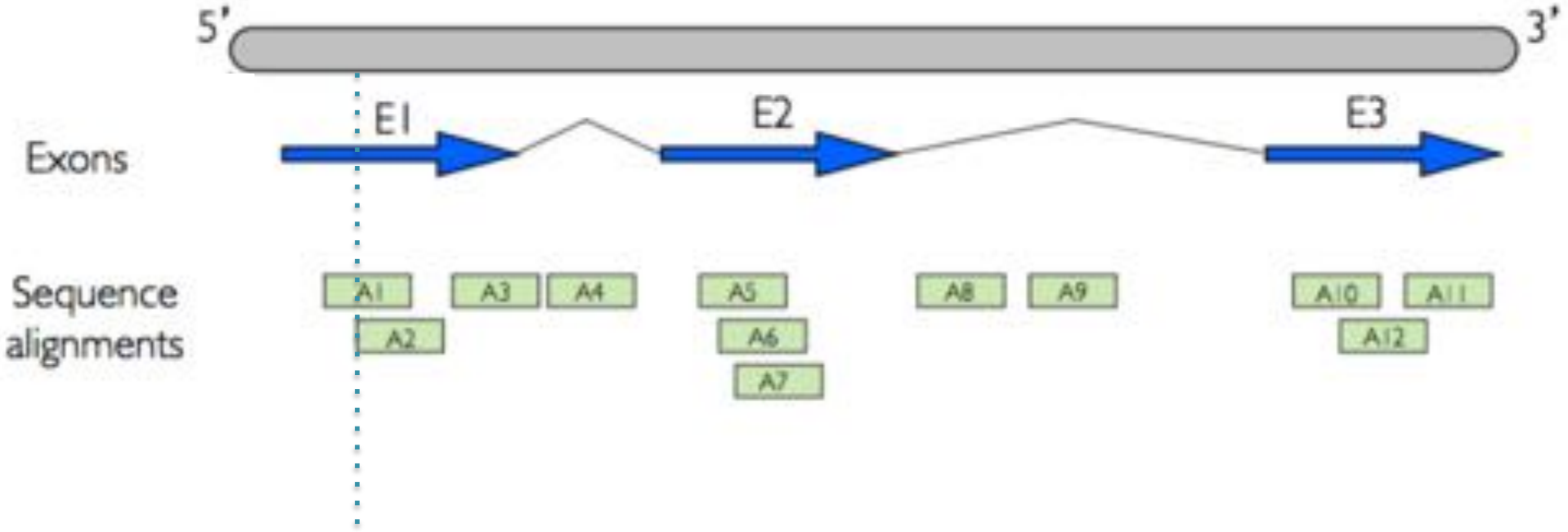
{E1}

Plane Sweep to the Rescue!



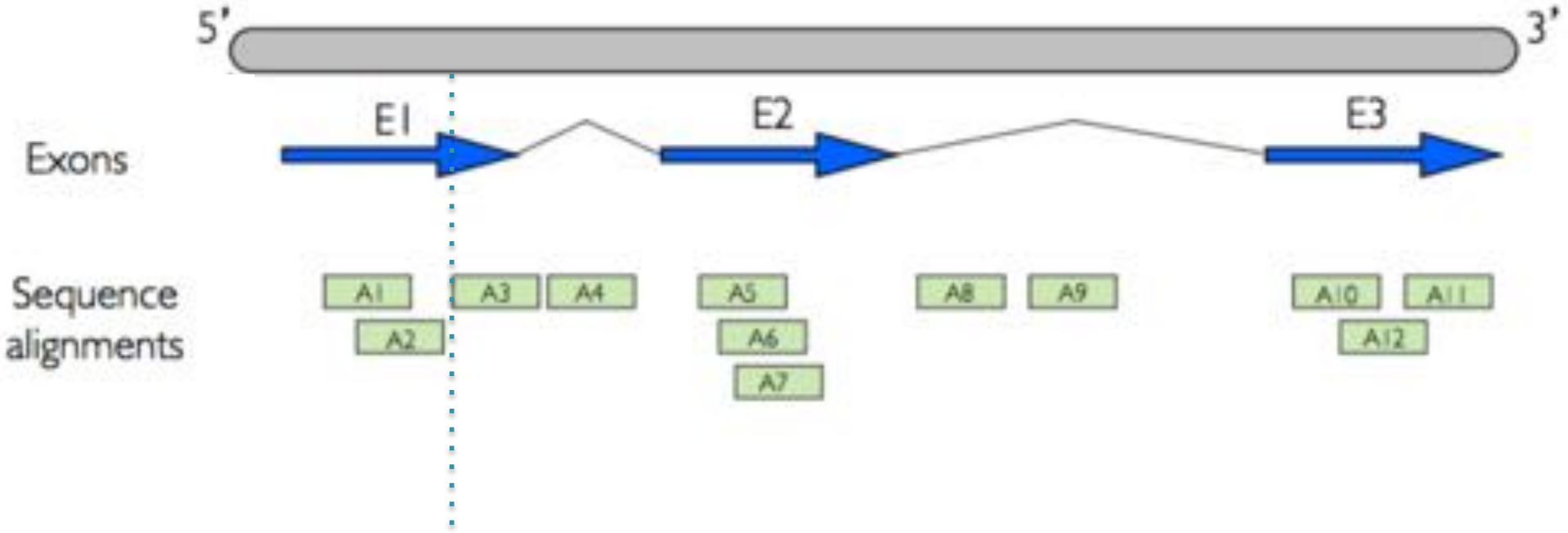

```
A1 overlaps E1

   {E1=(A1)}
```

Plane Sweep to the Rescue!



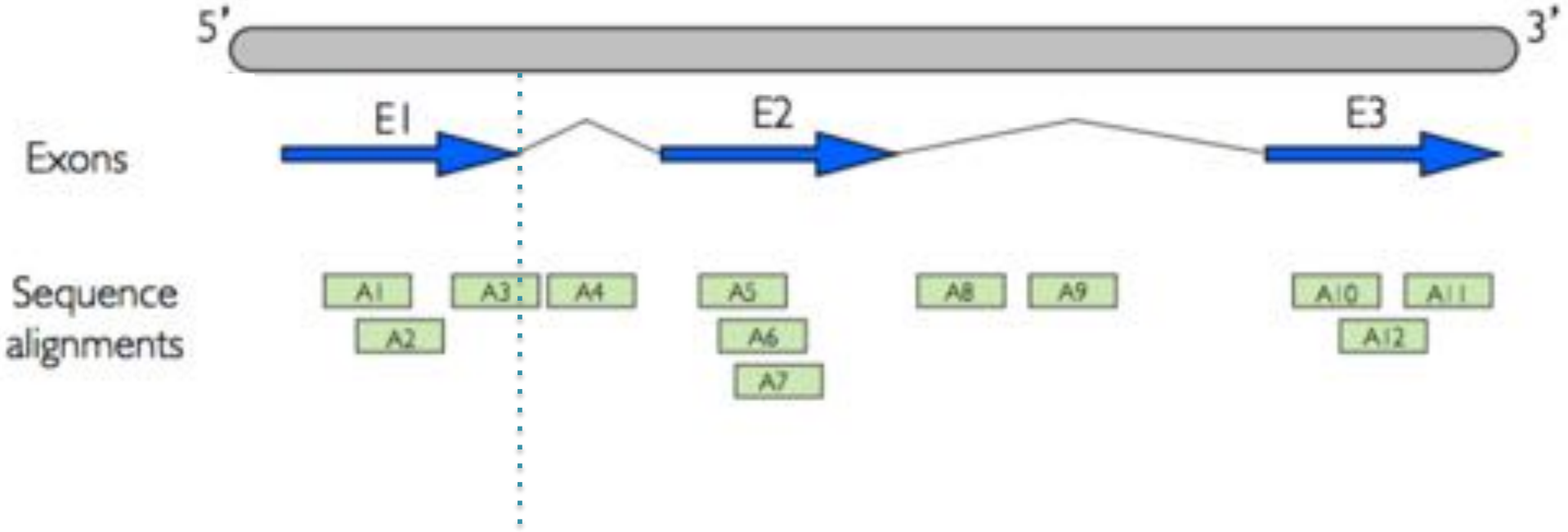

```
A2 overlaps E1

{E1=(A1, A2)}
```

Plane Sweep to the Rescue!




```
A3 overlaps E1

{E1=(A1, A2, A3)}
```

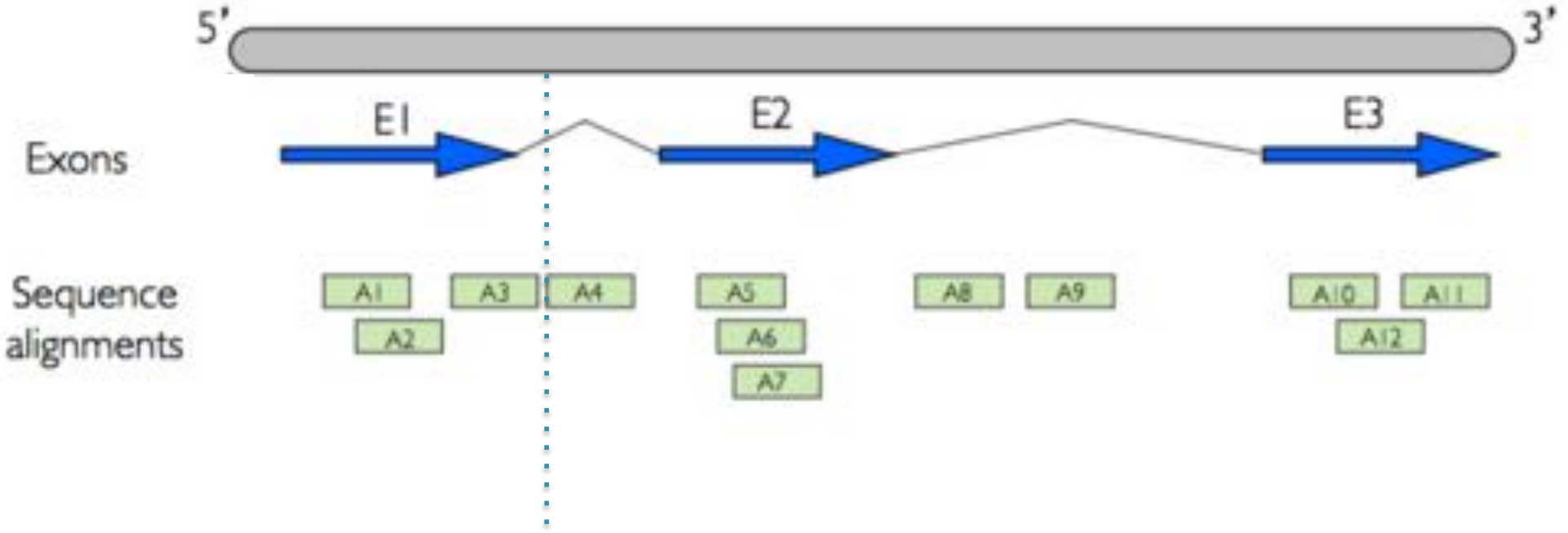
Plane Sweep to the Rescue!



End of E1

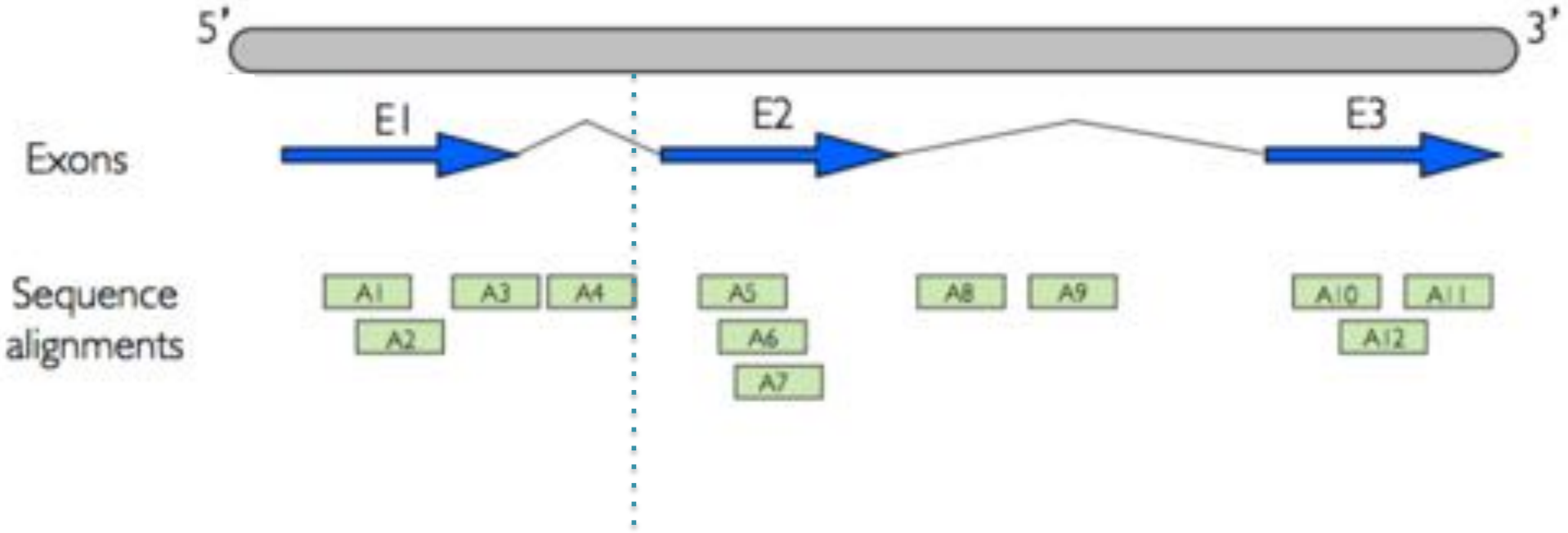
Report:
{E1=(A1, A2, A3)}

Plane Sweep to the Rescue!



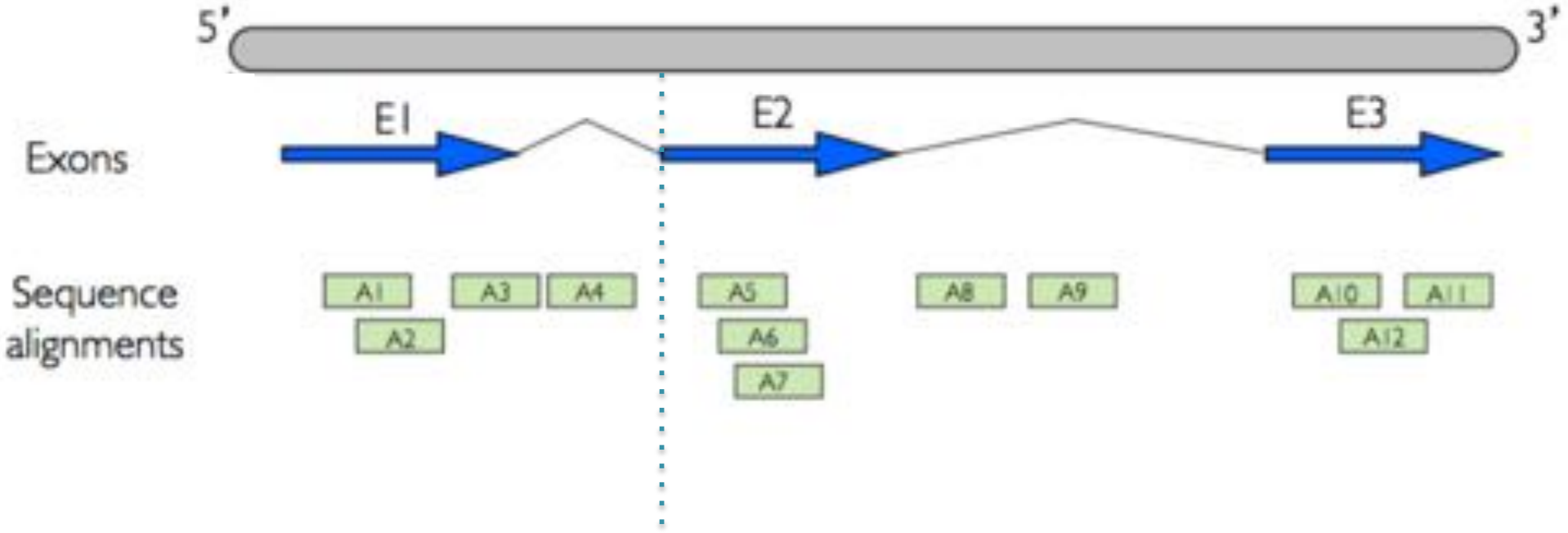
A4 starts,
but nothing is active

Plane Sweep to the Rescue!



A4 end,
but nothing is active

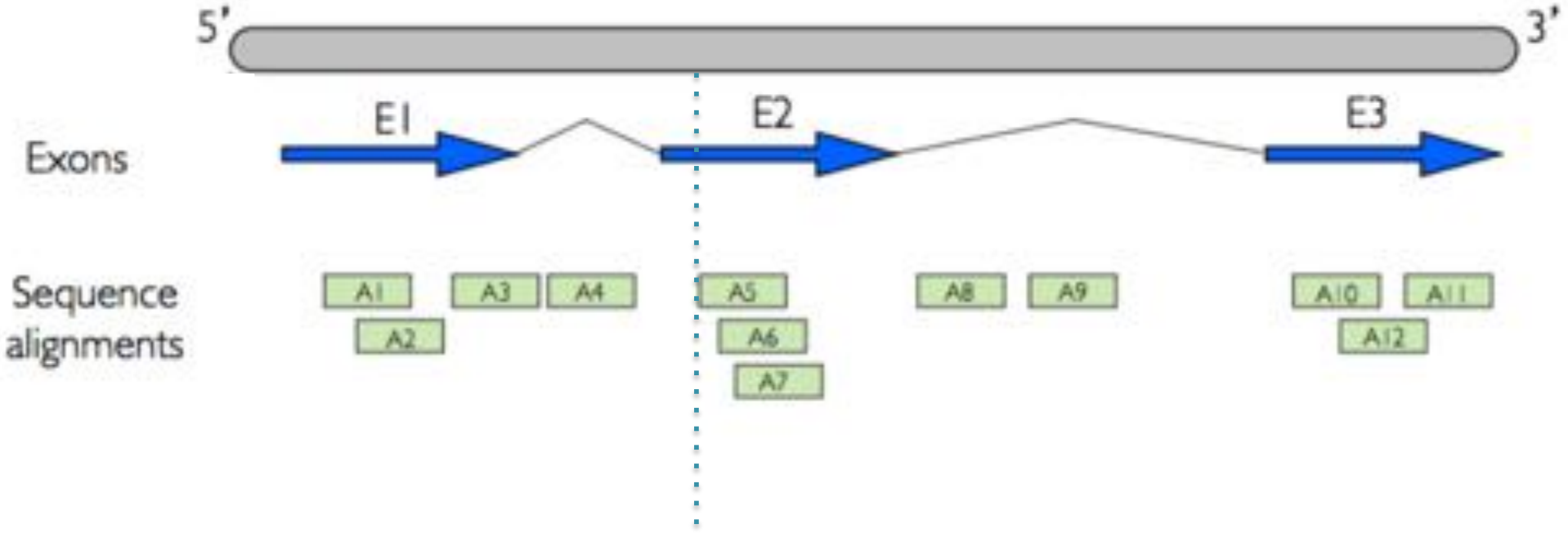
Plane Sweep to the Rescue!



E2 starts

{E2}

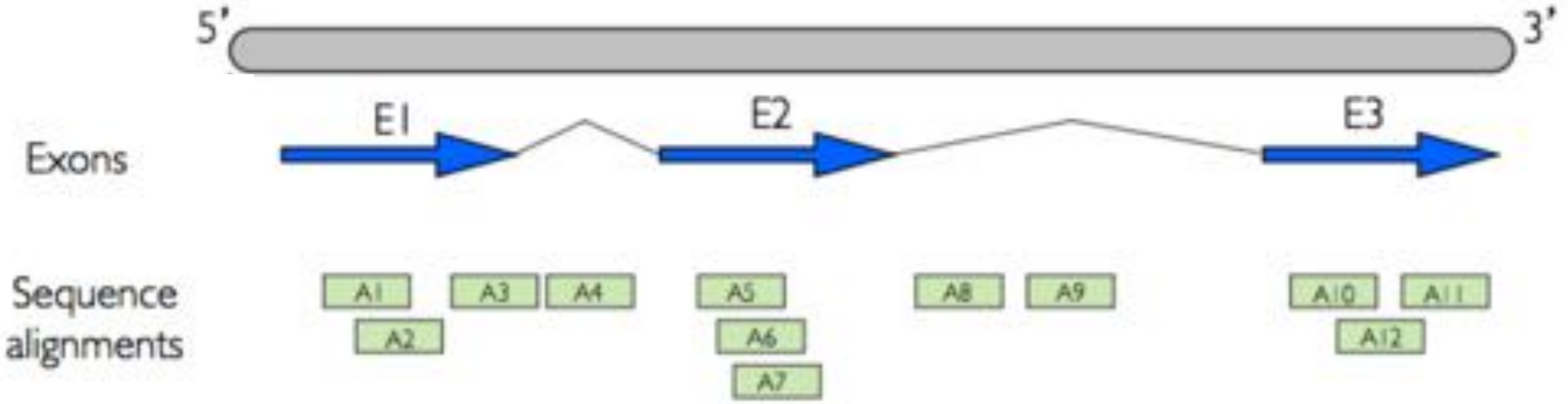
Plane Sweep to the Rescue!



A5 overlaps E2

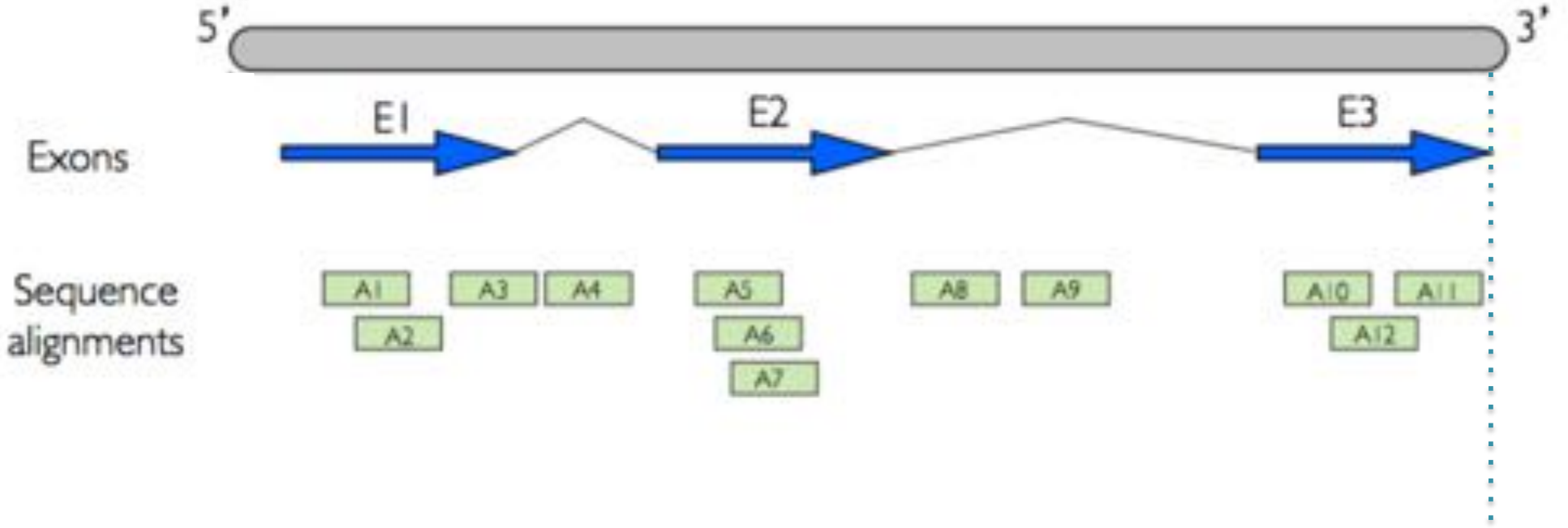
{E2=(A5)}

Plane Sweep to the Rescue!



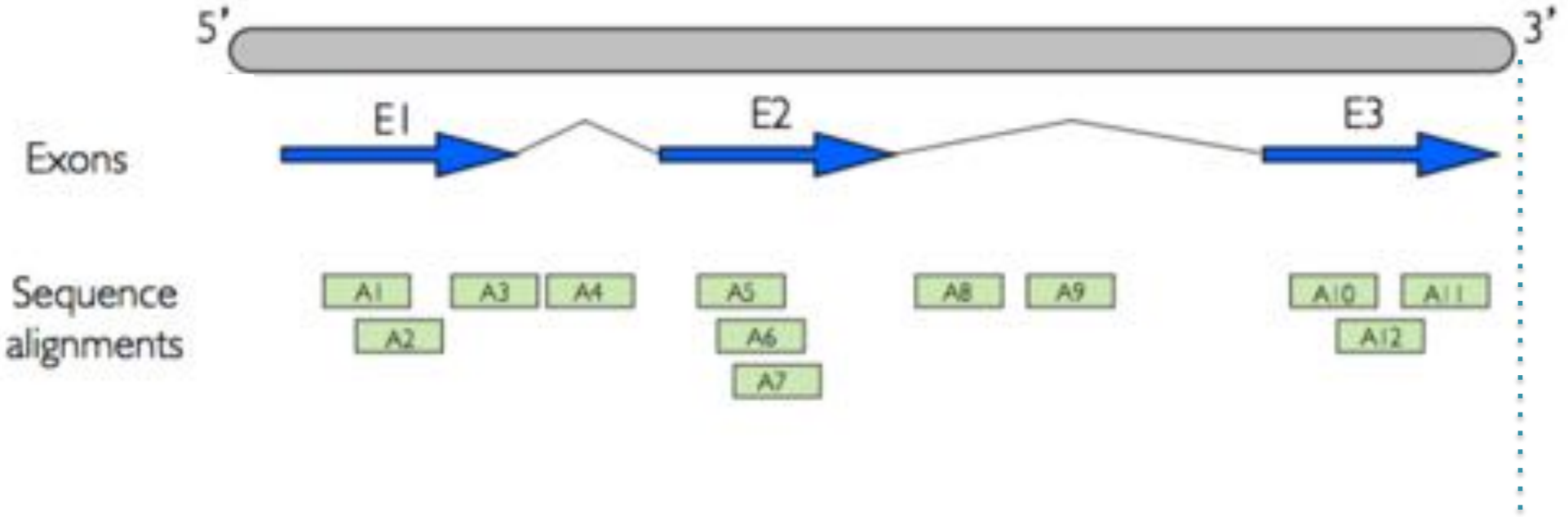
...

Plane Sweep to the Rescue!



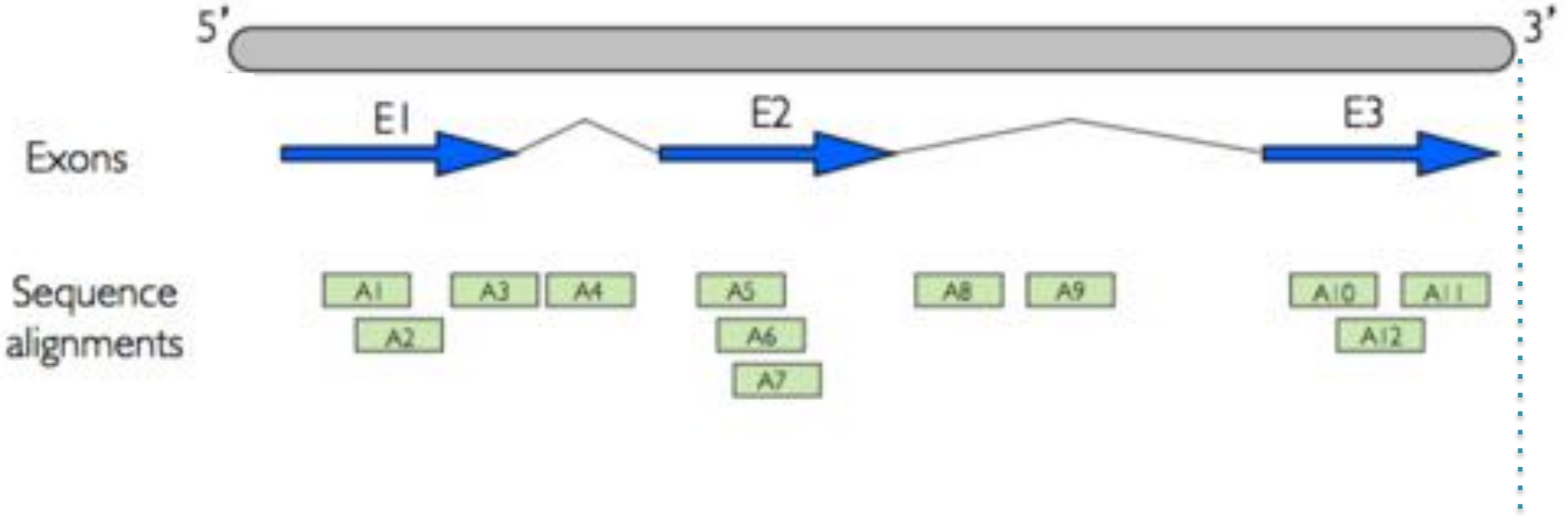
E3 Ends

Plane Sweep to the Rescue!



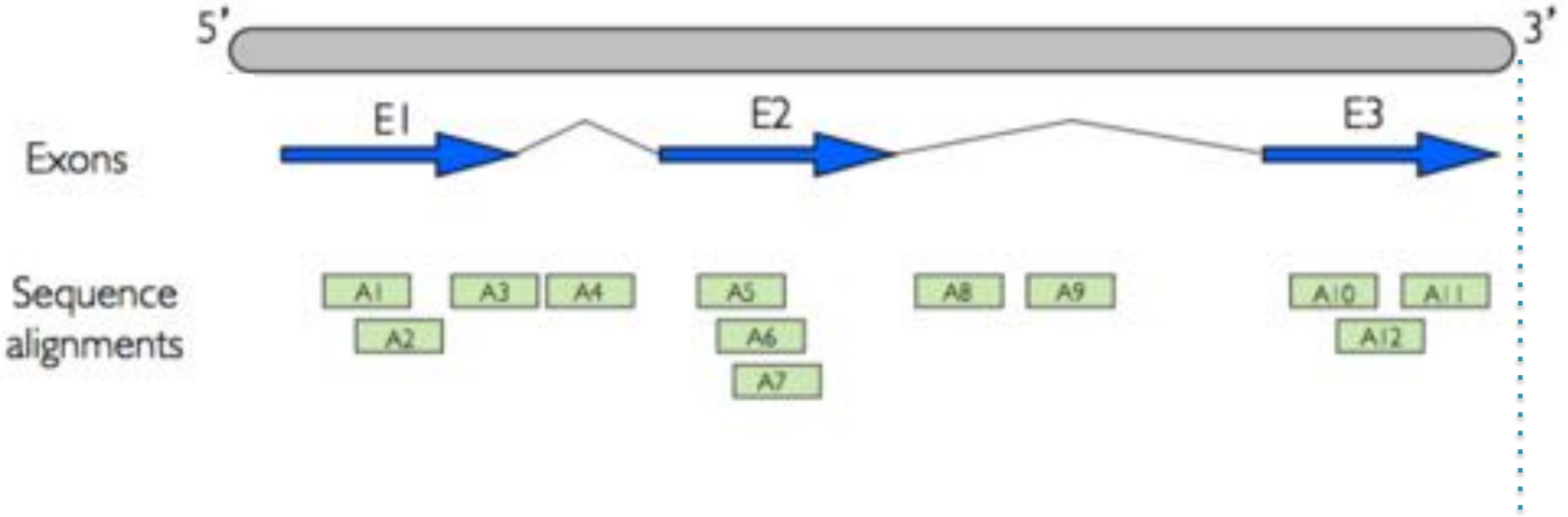
All done!

Plane Sweep to the Rescue!




```
Final Results:

E1=(A1,A2,A3)
E2=(A5,A6,A7)
E3=(A10,A12,A11)
```

Plane Sweep to the Rescue!



How many comparisons does the plane sweep algorithm make?

Each read is compared to the “active set”

Relatively few exons overlap: average ~1.1 active exons/position

Total comparisons: 900M reads * 1.1 “active exons/read” = 990M comparisons ☺

Heap-based Plane-Sweep

```
## record the delta encoded depth using a plane sweep
deltacovplane = []

## use a list to record the end positions of the elements currently in plane
planeheap = []

## BEGIN SWEEP (note change to index based so can peek ahead)
for rr in xrange(len(reads)):
  r = reads[rr]
  startpos = r[1]
  endpos   = r[2]

  ## clear out any positions from the plane that we have already moved past
  while (len(planeheap) > 0):

    if (planeheap[0] <= startpos):
      ## the coverage steps down, extract it from the front of the list
      ## oldend = planelist.pop(0)
      oldend = heapq.heappop(planeheap)

      nextend = -1
      if (len(planeheap) > 0):
        nextend = planeheap[0]

      ## only record this transition if it is not the same as a start pos
      ## and only if not the same as the next end point
      if ((oldend != startpos) and (oldend != nextend)):
        deltacovplane.append((oldend, len(planeheap)))
    else:
      break

  ## Now insert the current endpos into the correct position into the list
  heapq.heappush(planeheap, endpos)


  ## Finally record that the coverage has increased
  ## But make sure the current read does not start at the same position as the next
  if ((rr == len(reads)-1) or (startpos != reads[rr+1][1])):
    deltacovplane.append((startpos, len(planeheap)))

  ## if it is at the same place, it will get reported in the next cycle


## Flush any remaining end positions
while (len(planeheap) > 0):
  ##oldend = planelist.pop(0)
  oldend = heapq.heappop(planeheap)
  deltacovplane.append((oldend, len(planeheap)))
```

Keep track of reads "touching" the sweep plane using a heap

Compare the new read startpos to the smallest endpos on the sweep

If we have moved past a read, remove from heap

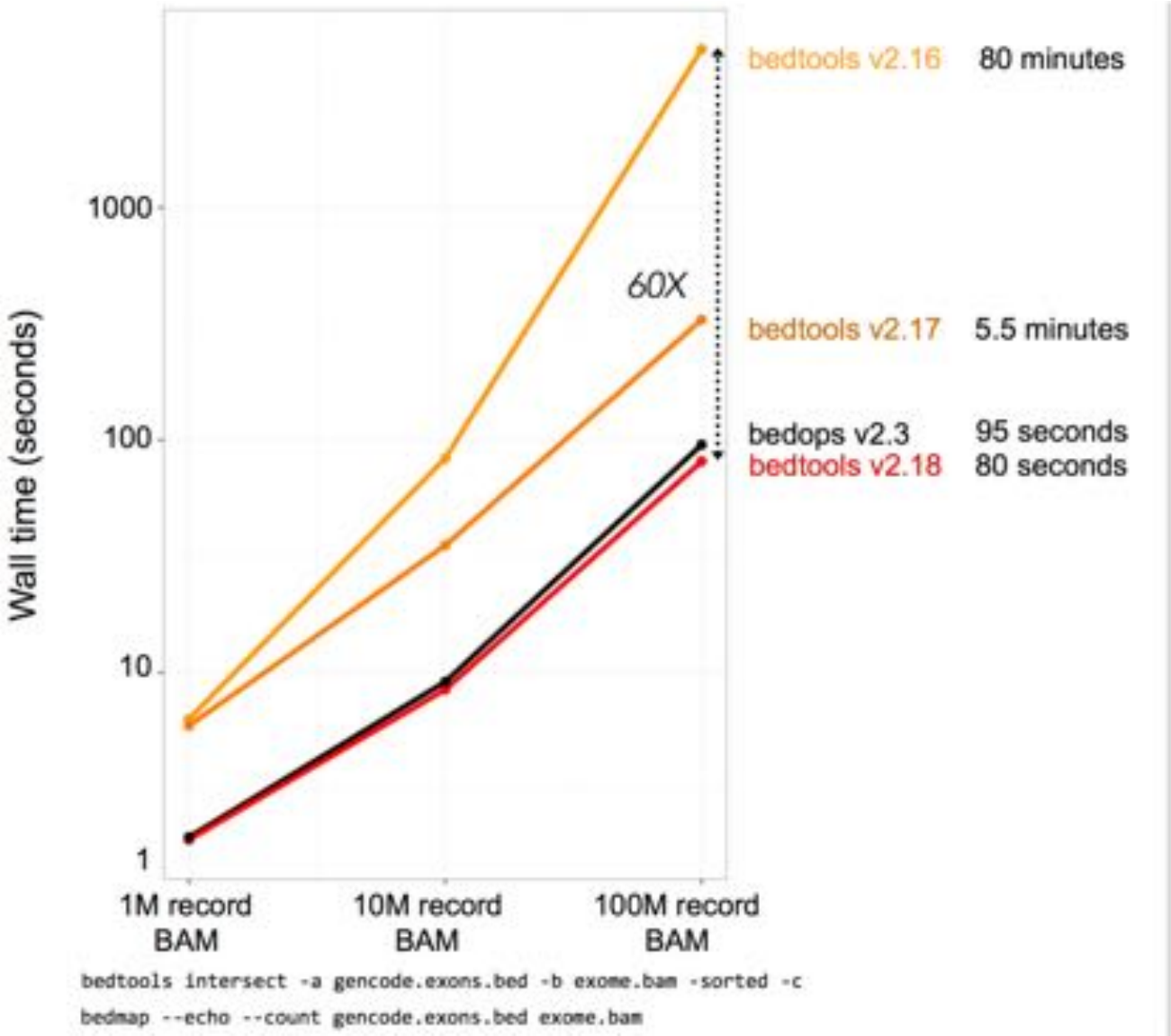
Slightly annoying bookkeeping for reads with same coordinates

Now add the new read to the sweep plane

Flush at the end

```
Beginning heap-based plane sweep over 1873 reads
Heap-Plane sweep found 3698 steps, saving 99.62% of the space in 14.26 ms (311.08 speedup)!
```

BEDTools Performance



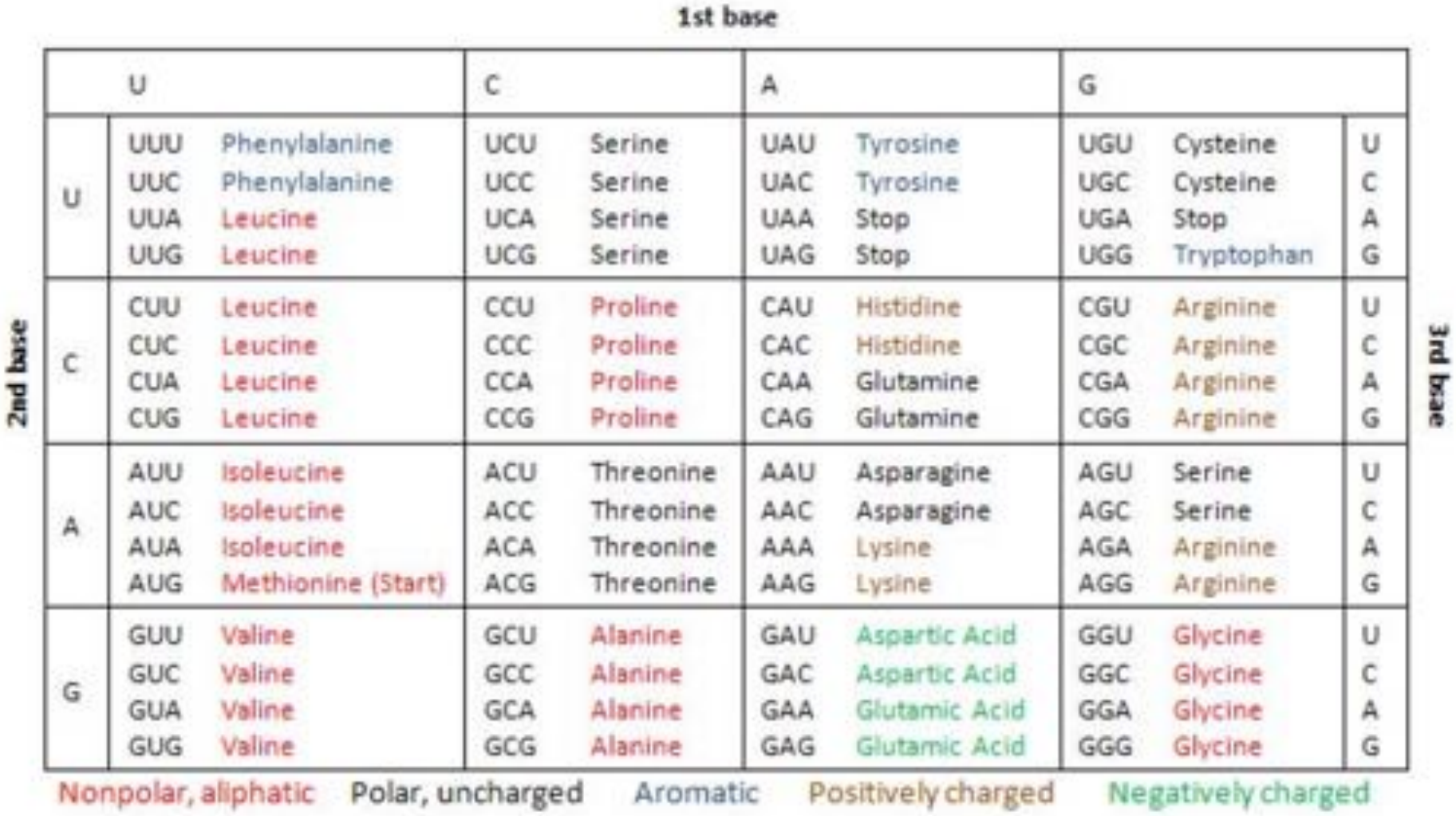
Part 2:
Genome Annotation

Goal: Genome Annotations

aatgcatgcggctatgctaatgcatgcggctatgctaagctgggatccgatgacaatgcatgcggctatgctaat
gcatgcggctatgcaagctgggatccgatgactatgctaagctgggatccgatgacaatgcatgcggctatgct
aatgaatggtcttgggatttaccttggaatgctaagctgggatccgatgacaatgcatgcggctatgctaatgaa
tggtcttgggatttaccttggaatatgctaatgcatgcggctatgctaagctgggatccgatgacaatgcatgcg
gctatgctaatgcatgcggctatgcaagctgggatccgatgactatgctaagctgcggctatgctaatgcatgcg
gctatgctaagctgggatccgatgacaatgcatgcggctatgctaatgcatgcggctatgcaagctgggatcct
gcggctatgctaatgaatggtcttgggatttaccttggaatgctaagctgggatccgatgacaatgcatgcggct
atgctaatgaatggtcttgggatttaccttggaatatgctaatgcatgcggctatgctaagctgggaatgcatgcg
gctatgctaagctgggatccgatgacaatgcatgcggctatgctaatgcatgcggctatgcaagctgggatccg
atgactatgctaagctgcggctatgctaatgcatgcggctatgctaagctcatgcggctatgctaagctgggaat
gcatgcggctatgctaagctgggatccgatgacaatgcatgcggctatgctaatgcatgcggctatgcaagctg
ggatccgatgactatgctaagctgcggctatgctaatgcatgcggctatgctaagctcggctatgctaatgaatg
gtcttgggatttaccttggaatgctaagctgggatccgatgacaatgcatgcggctatgctaatgaatggtcttgg
gatttaccttggaatatgctaatgcatgcggctatgctaagctgggaatgcatgcggctatgctaagctgggatc
cgatgacaatgcatgcggctatgctaatgcatgcggctatgcaagctgggatccgatgactatgctaagctgcg
gctatgctaatgcatgcggctatgctaagctcatgcgg

Goal: Genome Annotations

Genetic Code



Outline

1. Alignment to other genomes
2. Prediction aka "Gene Finding"
3. Experimental & Functional Assays

Outline

1. Alignment to other genomes
2. Prediction aka "Gene Finding"
3. Experimental & Functional Assays

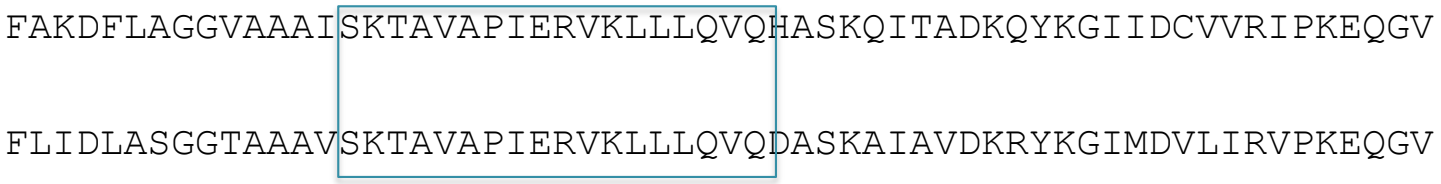
Basic Local Alignment Search Tool

- Rapidly compare a sequence Q to a database to find all sequences in the database with an score above some cutoff S.
  - Which protein is most similar to a newly sequenced one?
  - Where does this sequence of DNA originate?

- Speed achieved by using a procedure that typically finds "most" matches with scores > S.
  - Tradeoff between sensitivity and specificity/speed
    - Sensitivity – ability to find all related sequences
    - Specificity – ability to reject unrelated sequences

(Altschul et al. 1990)

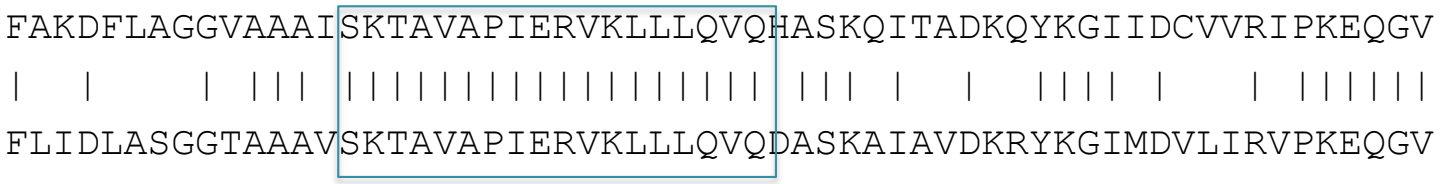
Seed and Extend



- Homologous sequences are likely to contain a short high scoring word pair, a seed.
  - Smaller seed sizes make the sense more sensitive, but also (much) slower
  - Typically do a fast search for prototypes, but then most sensitive for final result

- BLAST then tries to extend high scoring word pairs to compute high scoring segment pairs (HSPs).
  - Significance of the alignment reported via an e-value

Seed and Extend



- Homologous sequences are likely to contain a short high scoring word pair, a seed.
  - Smaller seed sizes make the sense more sensitive, but also (much) slower
  - Typically do a fast search for prototypes, but then most sensitive for final result

- BLAST then tries to extend high scoring word pairs to compute high scoring segment pairs (HSPs).
  - Significance of the alignment reported via an e-value

BLAST E-values

E-value = the number of HSPs having alignment score S (or higher) expected to occur by chance.

→ Smaller E-value, more significant in statistics

→ Bigger E-value, less significant

→ Over 1 means expect this totally by chance

(not significant at all!)

The expected number of HSPs with the score at least S is :

$$E = K*n*m*e^{-\lambda S}$$

K, λ are constant depending on model

n, m are the length of query and sequence

E-values quickly drop off for better alignment bits scores

Very Similar Sequences

```
Query: HBA_HUMAN Hemoglobin alpha subunit
Sbjct: HBB_HUMAN Hemoglobin beta subunit

Score =  114 bits (285),  Expect = 1e-26
Identities = 61/145 (42%), Positives = 86/145 (59%), Gaps = 8/145 (5%)

Query  2   LSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHF------DLSHGSAQV 55
           L+P +K+ V A WGKV  +  E G EAL R+ + +P T+ +F  F      D   G+ +V
Sbjct  3   LTPEEKSAVTALWGKV--NVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNPKV 60

Query  56  KGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAAHLPA 115
           K HGKKV  A ++ +AH+D++    + LS+LH  KL VDP NF+LL + L+  LA H
Sbjct  61  KAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKLHVDPENFRLLGNVLVCVLAHHFGK 120

Query  116 EFTPAVHASLDKFLASVSTVLTSKY 140
           EFTP V A+  K +A V+  L  KY
Sbjct  121 EFTPPVQAAYQKVVAGVANALAHKY 145
```

Quite Similar Sequences

```
Query: HBA_HUMAN Hemoglobin alpha subunit
Sbjct: MYG_HUMAN Myoglobin

Score = 51.2 bits (121), Expect = 1e-07,
Identities = 38/146 (26%), Positives = 58/146 (39%), Gaps = 6/146 (4%)

Query  2   LSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHF------DLSHGSAQV  55
           LS  +   V   WGKV A    +G E L R+F   P T   F  F      D    S  +
Sbjct  3   LSDGEWQLVLNVWGKVEADIPGHGQEVLIRLFKGHPETLEKFDKFKHLKSEDEMKASEDL  62

Query  56  KGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAAHLPA  115
           K HG  V  AL   +          +  L+  HA K ++     + +S C++  L +  P
Sbjct  63  KKHGATVLTALGGILKKKGHHEAEIKPLAQSHATKHKIPVKYLEFISECIIQVLQSKHPG  122

Query  116 EFTPAVHASLDKFLASVSTVLTSKYR  141
            +F      +++K L      + S Y+
Sbjct  123 DFGADAQGAMNKALELFRKDMASNYK  148
```

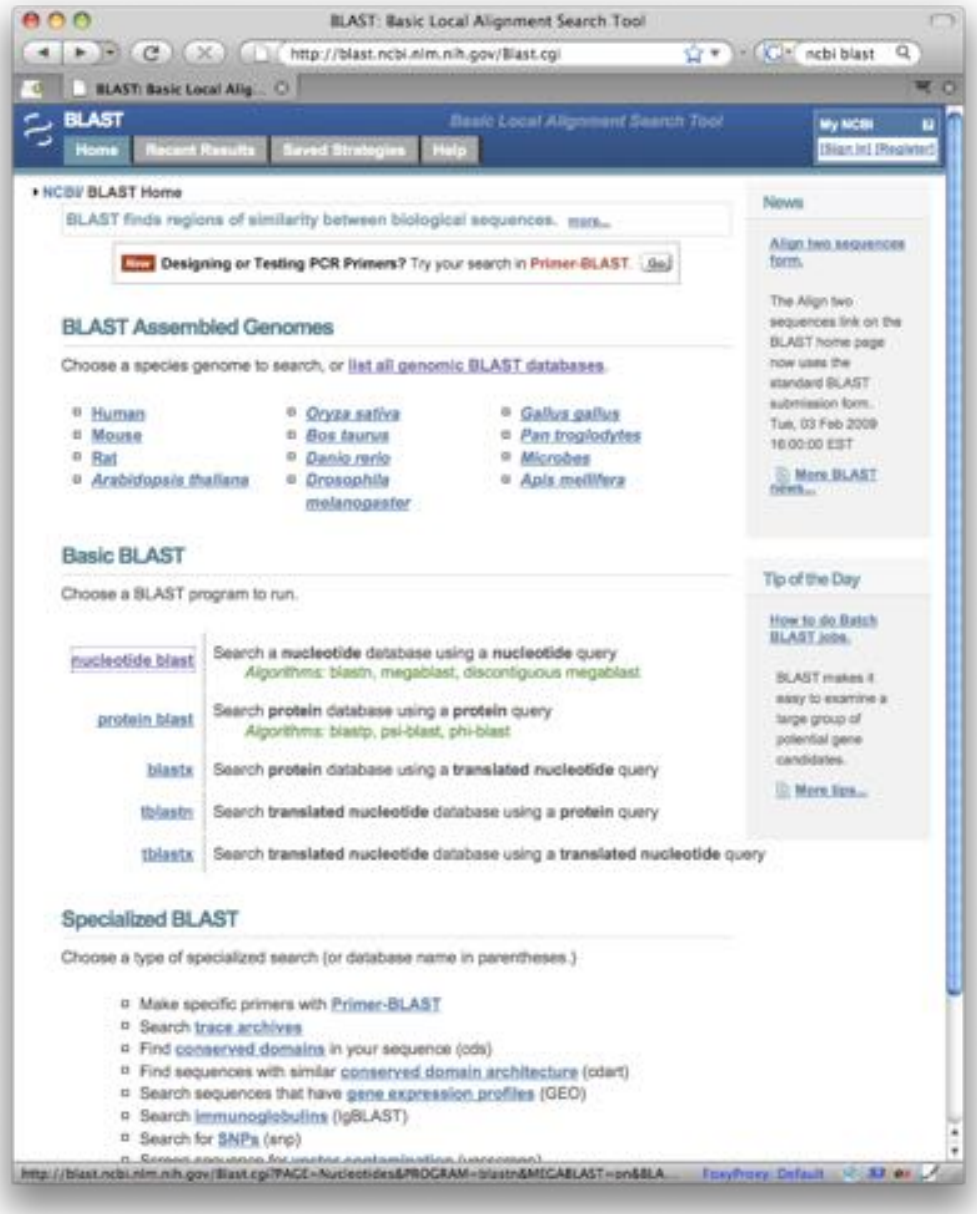
Not similar sequences

```
Query: HBA_HUMAN Hemoglobin alpha subunit
Sbjct: SPAC869.02c [Schizosaccharomyces pombe]

 Score = 33.1 bits (74),  Expect = 0.24
 Identities = 27/95 (28%), Positives = 50/95 (52%), Gaps = 10/95 (10%)

Query  30  ERMFLSFPTTKTYFPHFDLSHGSAQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAH  89
           ++M  ++P      P+F+ +H  +      + +A AL N   ++DD+  +LSA  D
Sbjct  59  QKMLGNYPEV---LPYFNKAHQISL--SQPRILAFALLNYAKNIDDL-TSLSAFMDQIVV 112

Query  90  K---LRVDPVNFKLLSHCLLVTLAAHLPAEF-TPA  120
           K   L++   ++ ++ HCLL T+   LP++  TPA
Sbjct  113 KHVGLQIKAEHYPIVGHCLLSTMQELLPSDVATPA  147
```

Blast Versions

Program	Database	Query
BLASTN	Nucleotide	Nucleotide
BLASTP	Protein	Protein
BLASTX	Protein	Nucleotide translated into protein
TBLASTN	Nucleotide translated into protein	Protein
TBLASTX	Nucleotide translated into protein	Nucleotide translated into protein

NCBI Blast



- Nucleotide Databases
  - nr: All Genbank
  - refseq: Reference organisms
  - wgs: All reads

- Protein Databases
  - nr: All non-redundant sequences
  - Refseq: Reference proteins

Outline

1. Alignment to other genomes
2. Prediction aka "Gene Finding"
3. Experimental & Functional Assays

Bacterial Gene Finding and Glimmer

(also Archaeal and viral gene finding)

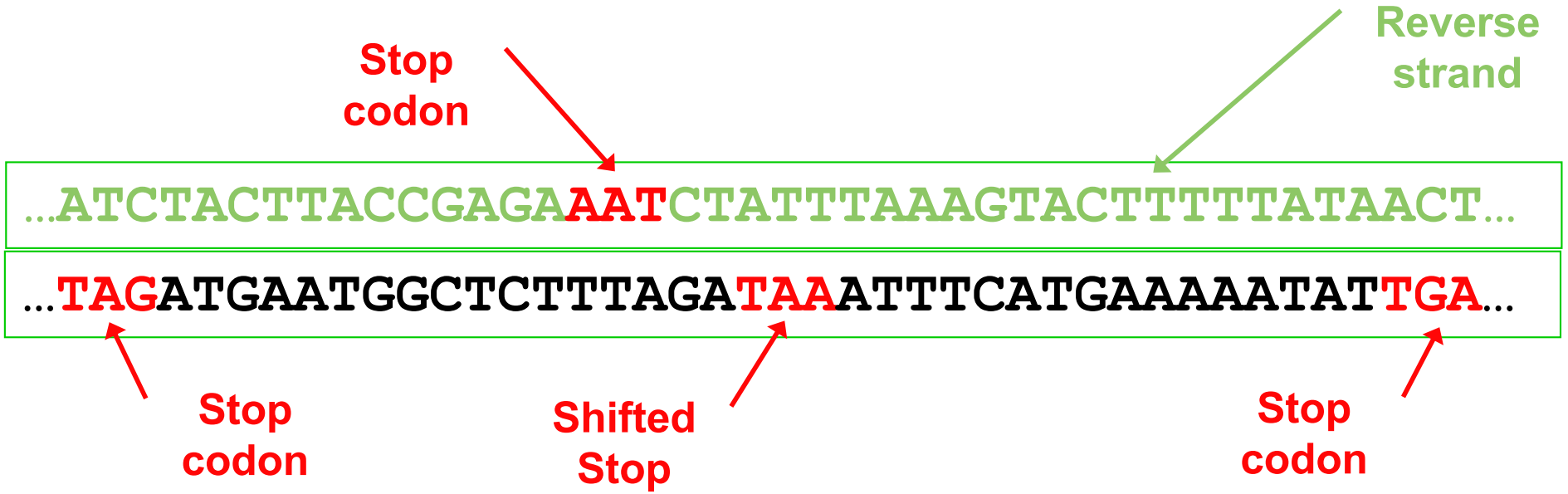
Arthur L. Delcher and Steven Salzberg
Center for Bioinformatics and Computational Biology
Johns Hopkins University School of Medicine

Step One

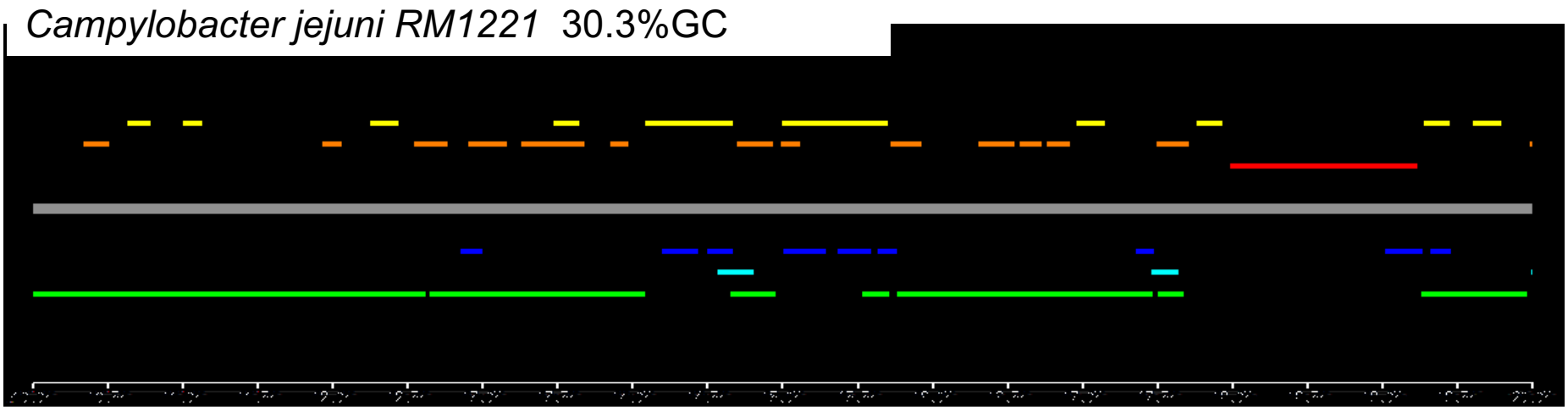
- Find open reading frames (ORFs).

Step One

- Find open reading frames (ORFs).



- But ORFs generally overlap …

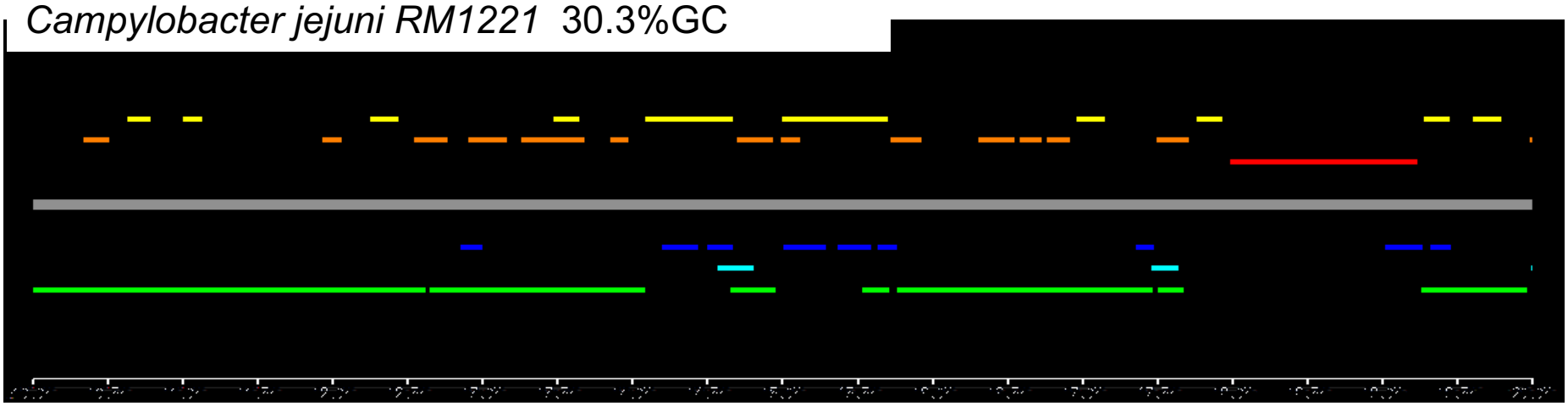


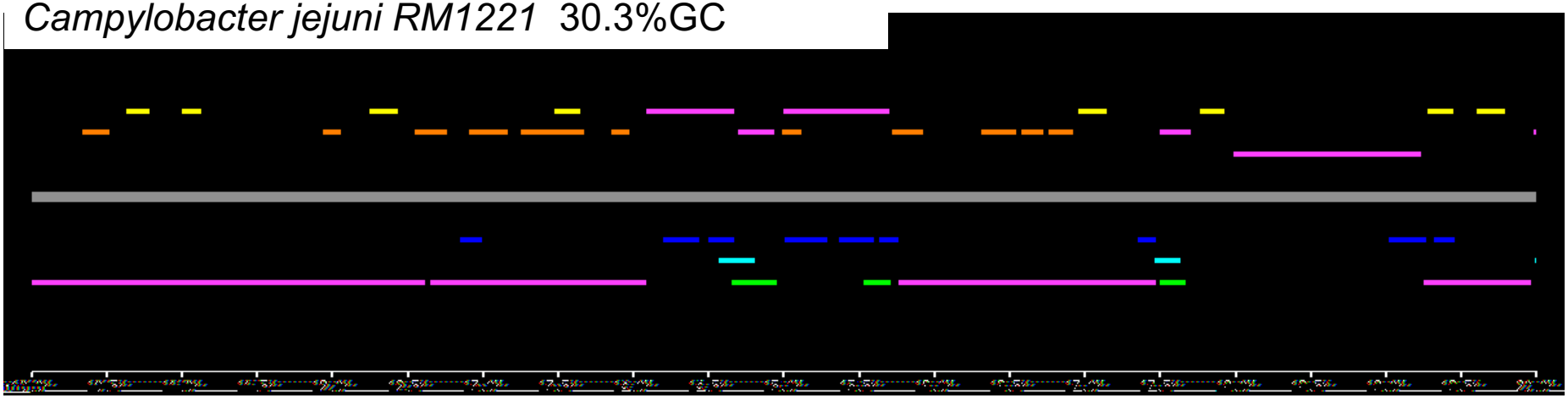
All ORFs longer than 100bp on both strands shown
- color indicates reading frame

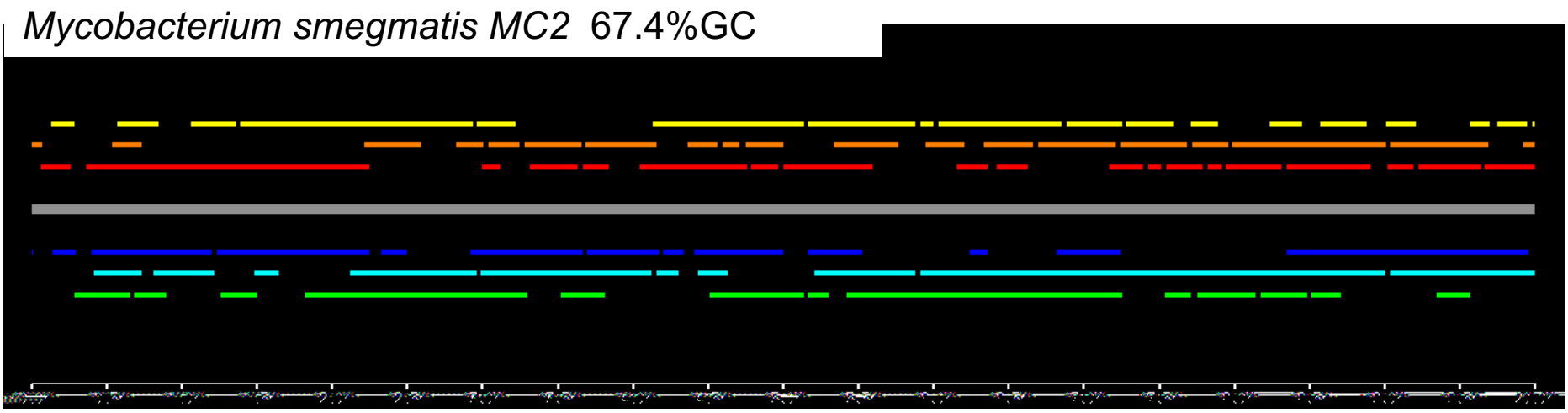
Longest ORFs likely to be protein-coding genes

Note the low GC content

All genes are ORFs but not all ORFs are genes

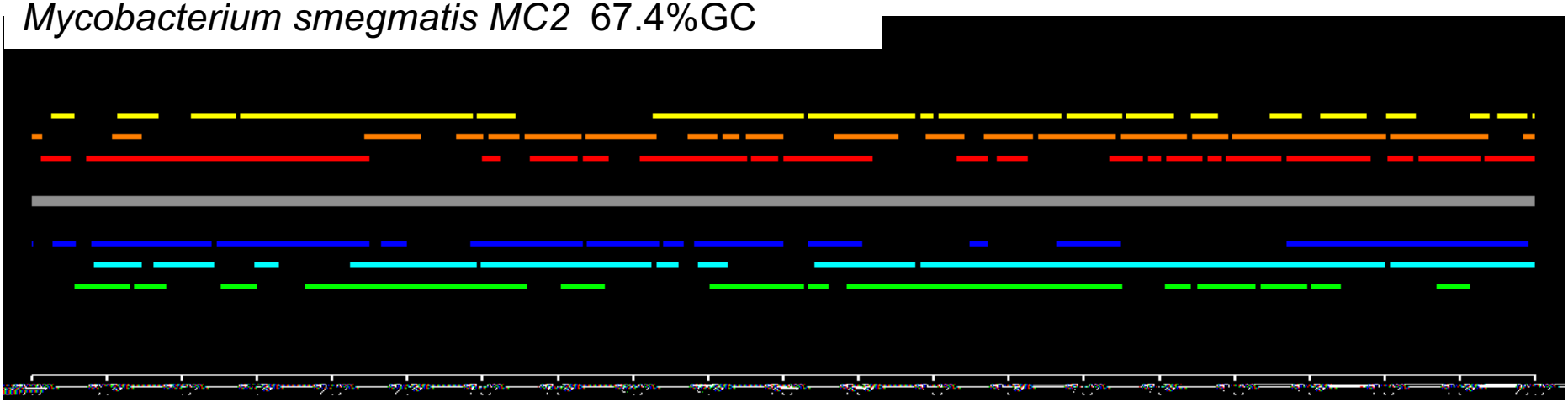

Campylobacter jejuni RM1221 30.3%GC


Campylobacter jejuni RM1221 30.3%GC

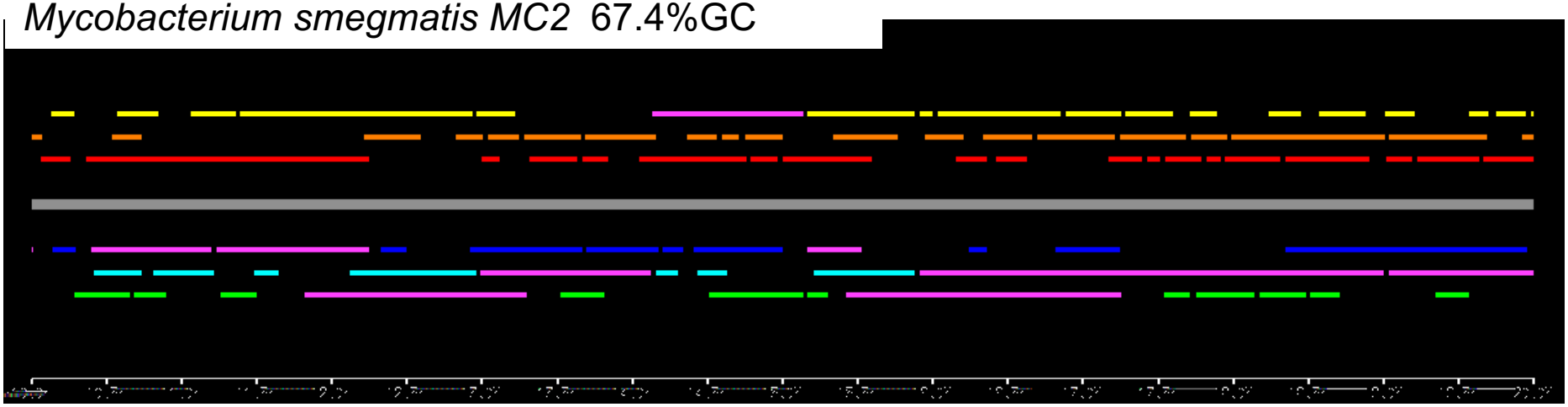


Note what happens in a high-GC genome

Mycobacterium smegmatis MC2 67.4%GC



Mycobacterium smegmatis MC2 67.4%GC



Probabilistic Methods

- Create models that have a probability of generating any given sequence.
  - Evaluate gene/non-genome models against a sequence
- Train the models using examples of the types of sequences to generate.
  - Use RNA sequencing, homology, or "obvious" genes
- The "score" of an orf is the probability of the model generating it.
  - Most basic technique is to count how kmers occur in known genes versus intergenic sequences
  - More sophisticated methods consider variable length contexts, "wobble" bases, other statistical clues

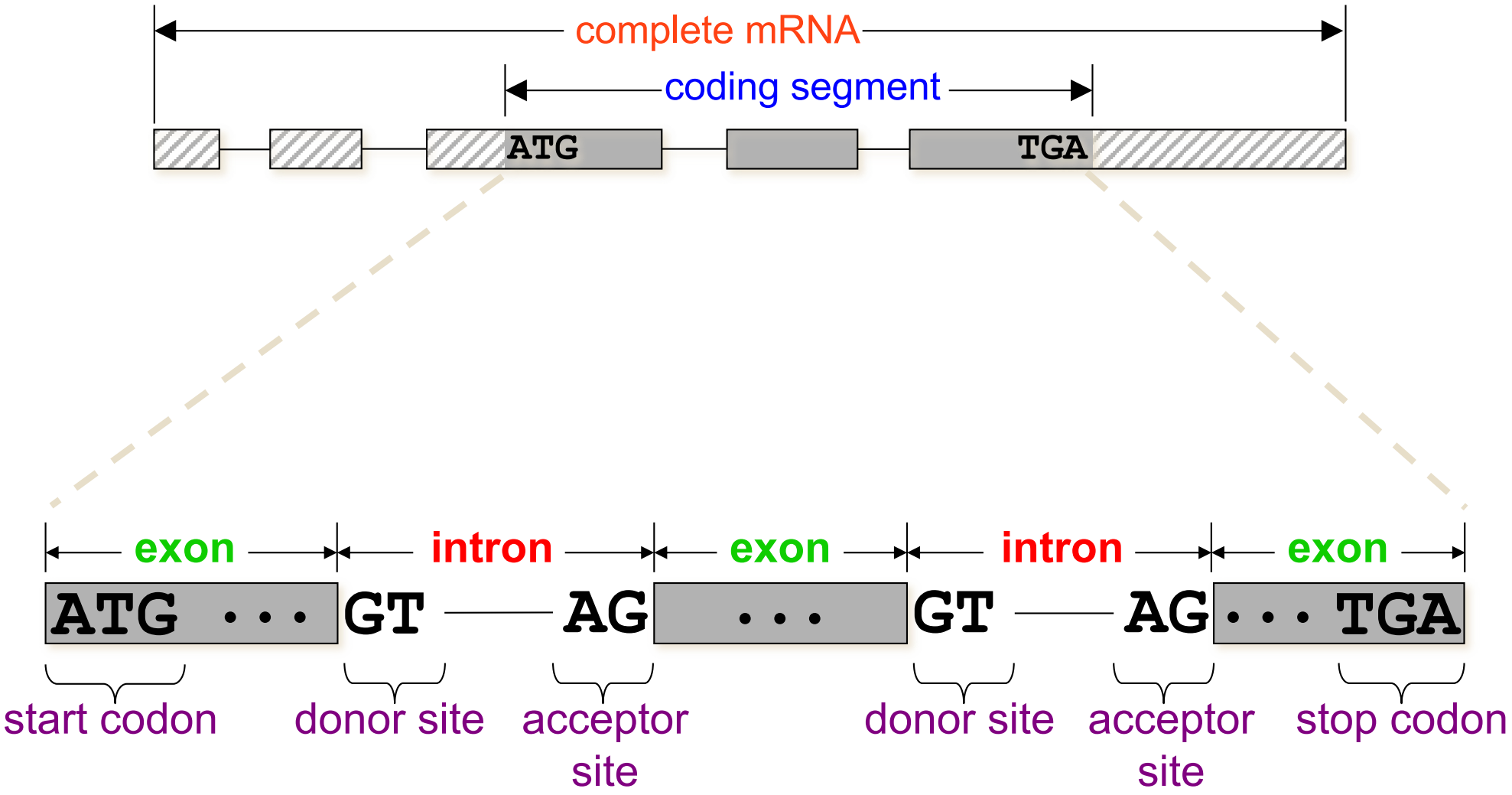
Overview of Eukaryotic Gene Prediction

CBB 231 / COMPSCI 261

W.H. Majoros

Duke UNIVERSITY

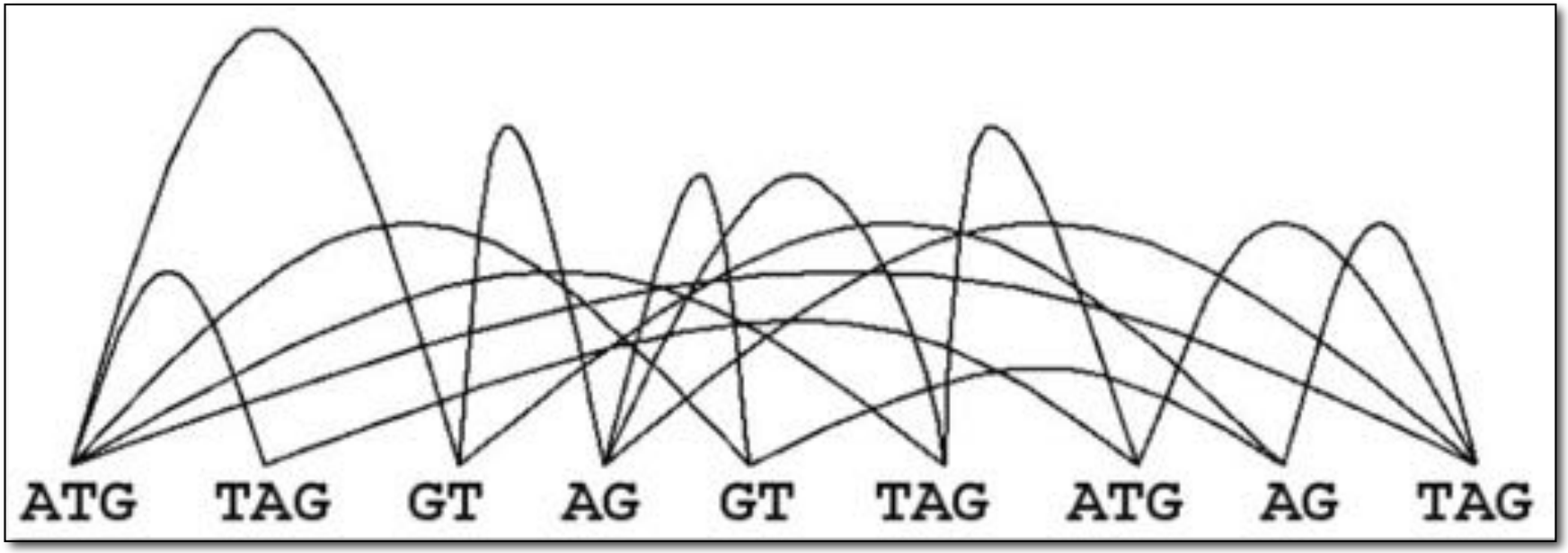
Eukaryotic Gene Syntax



Regions of the gene outside of the CDS are called *UTR*'s (*untranslated regions*), and are mostly ignored by gene finders, though they are important for regulatory functions.

Representing Gene Syntax with ORF Graphs

After identifying the most promising (i.e., highest-scoring) signals in an input sequence, we can apply the gene syntax rules to connect these into an *ORF graph*:

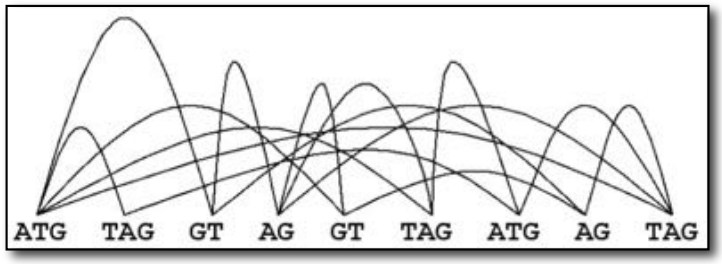


An ORF graph represents all possible *gene parses* (and their scores) for a given set of putative signals. A *path* through the graph represents a single gene parse.

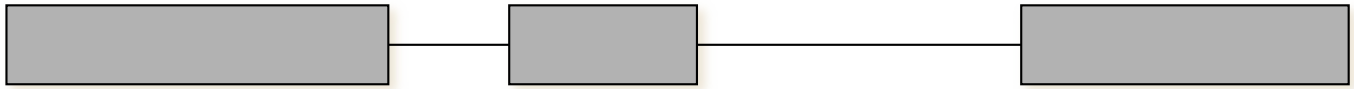
Conceptual Gene-finding Framework

```
TATTCCGATCGATCGATCTCTCTAGCGTCTACG
CTATCATCGCTCTCTATTATCGCGCGATCGTCG
ATCGCGCGAGAGTATGCTACGTCGATCGAATTG
```

identify most promising signals, score signals and content regions between them; induce an ORF graph on the signals

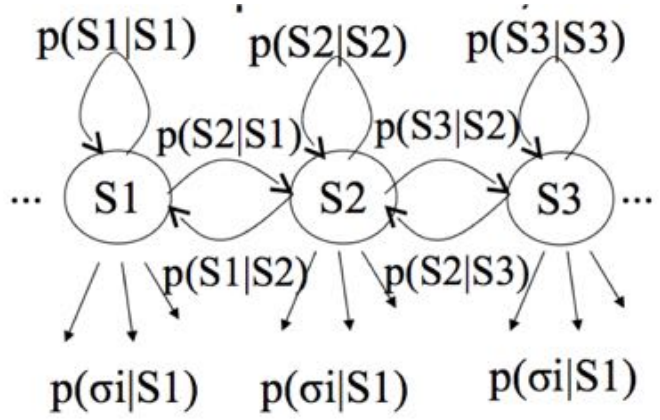


find highest-scoring path through ORF graph; interpret path as a gene parse = gene structure



Why Hidden?

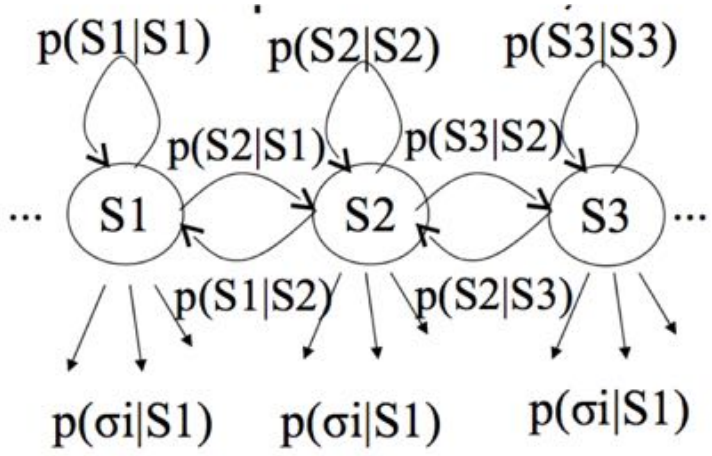
- Similar to Markov models used for prokaryotic gene finding, but system may transition between multiple models called states (gene/non-gene, intergenic/exon/intron)
- Observers can see the emitted symbols of an HMM (i.e., nucleotides) but have no ability to know which state the HMM is currently in.
  - But we can ***infer*** the most likely hidden states of an HMM based on the given sequence of emitted symbols.



AAAGCATGCATTTAACGTGAGCACAATAGATTACA

What is an HMM?

- Dynamic Bayesian Network
  - A set of states
    - {Fair, Biased} for coin tossing
    - {Gene, Not Gene} for Bacterial Gene
    - {Intergenic, Exon, Intron} for Eukaryotic Gene

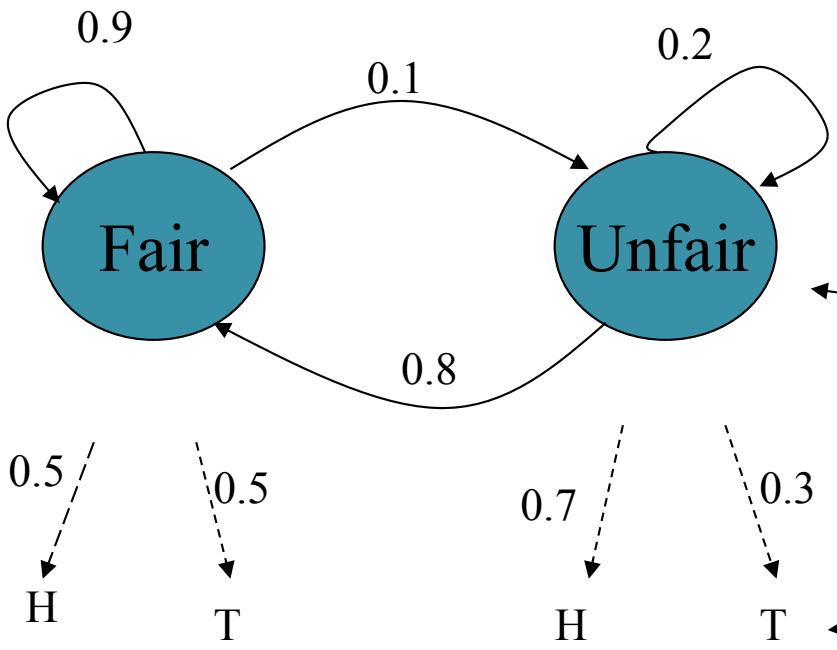


  - A set of emission characters
    - E={H,T} for coin tossing
    - E={1,2,3,4,5,6} for dice tossing
    - E={A,C,G,T} = for DNA

  - State-specific emission probabilities
    - P(H | Fair) = .5, P(T | Fair) = .5, P(H | Biased) = .9, P(T | Biased) = .1
    - P(A | Gene) = .9, P(A | Not Gene) = .1 …

  - A probability of taking a transition
    - $P(s_i=\text{Fair}|s_{i-1}=\text{Fair}) = .9$, $P(s_i=\text{Bias}|s_{i-1} = \text{Fair}) .1$
    - $P(s_i=\text{Exon} | s_{i-1}=\text{Intergenic})$, …

HMM Example - Casino Coin



State transition probs.

States

Symbol emission probs.

Observation Symbols

Observation Sequence

State Sequence

HTHHTTHHHTHTHTHHTHHHHHHTHTHH

FFFFFFUUUFFFFFFUUUUUUUFFFFFF

Motivation: Given a sequence of H & Ts, can you tell at what times the casino cheated?

Slide credit: Fatih Gelgi, Arizona State U.

Three classic HMM problems

1. **Evaluation**: given a model and an output sequence, what is the probability that the model generated that output?
2. **Decoding**: given a model and an output sequence, what is the most likely state sequence through the model that generated the output?
3. **Learning**: given a model and a set of observed sequences, how do we set the model's parameters so that it has a high probability of generating those sequences?

Three classic HMM problems

1. **Evaluation**: given a model and an output sequence, what is the probability that the model generated that output?

- To answer this, we consider all possible paths through the model

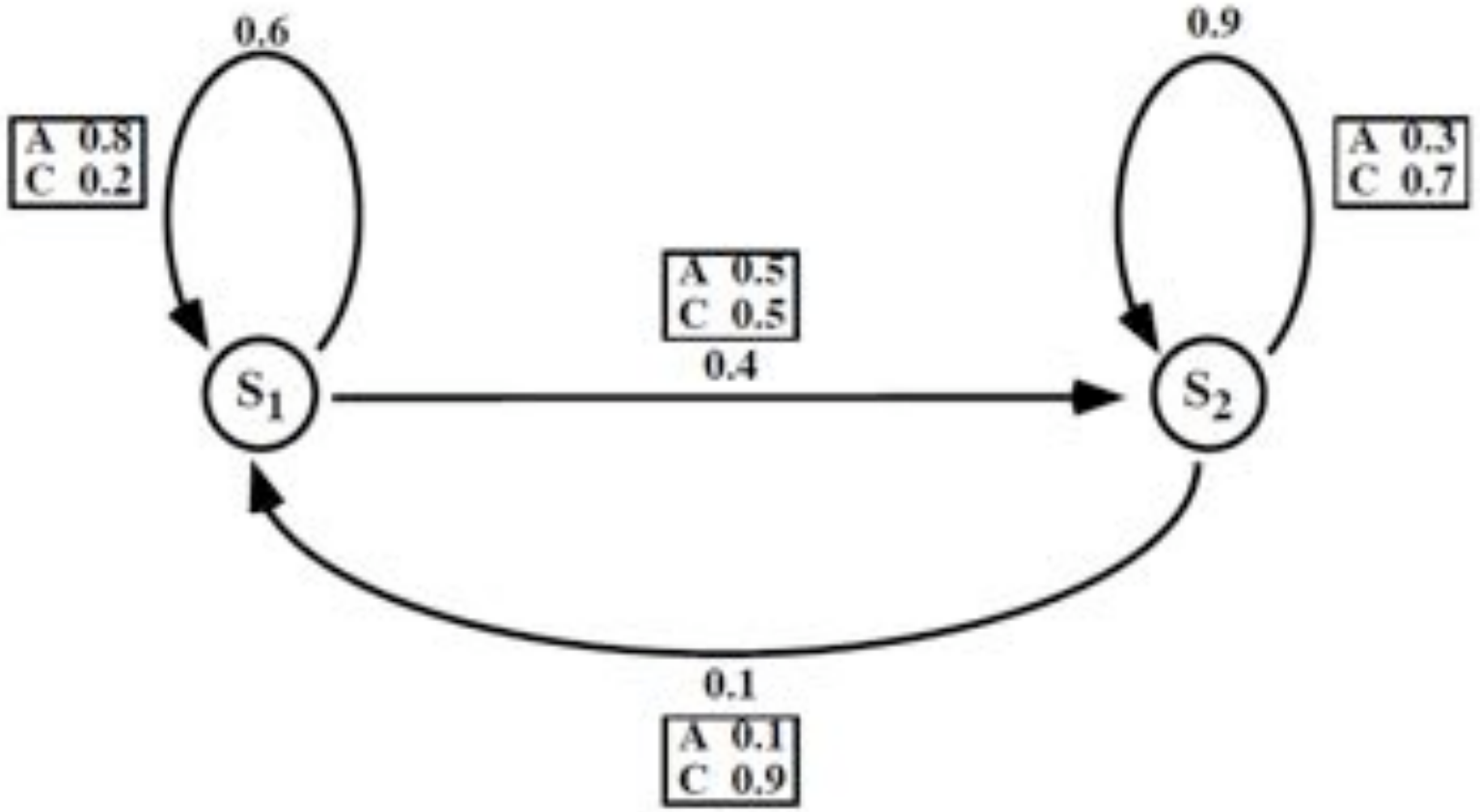
- Example: we might have a set of HMMs representing protein families -> pick the model with the best score

Solving the Evaluation problem: The Forward algorithm

- To solve the Evaluation problem (probability that the model generated the sequence), we use the HMM and the data to build a *trellis*

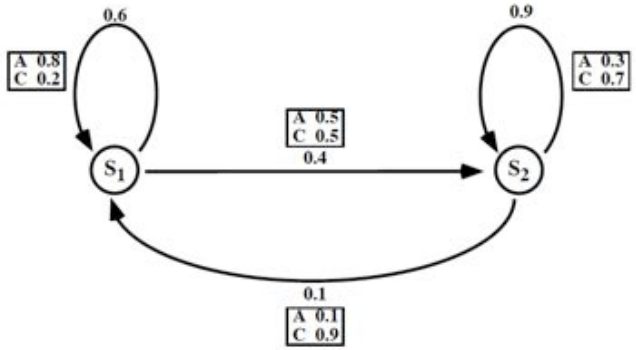
- Filling in the trellis will give tell us the probability that the HMM generated the data by finding all possible paths that could do it
  - Especially useful to evaluate from which models, a given sequence is most likely to have originated

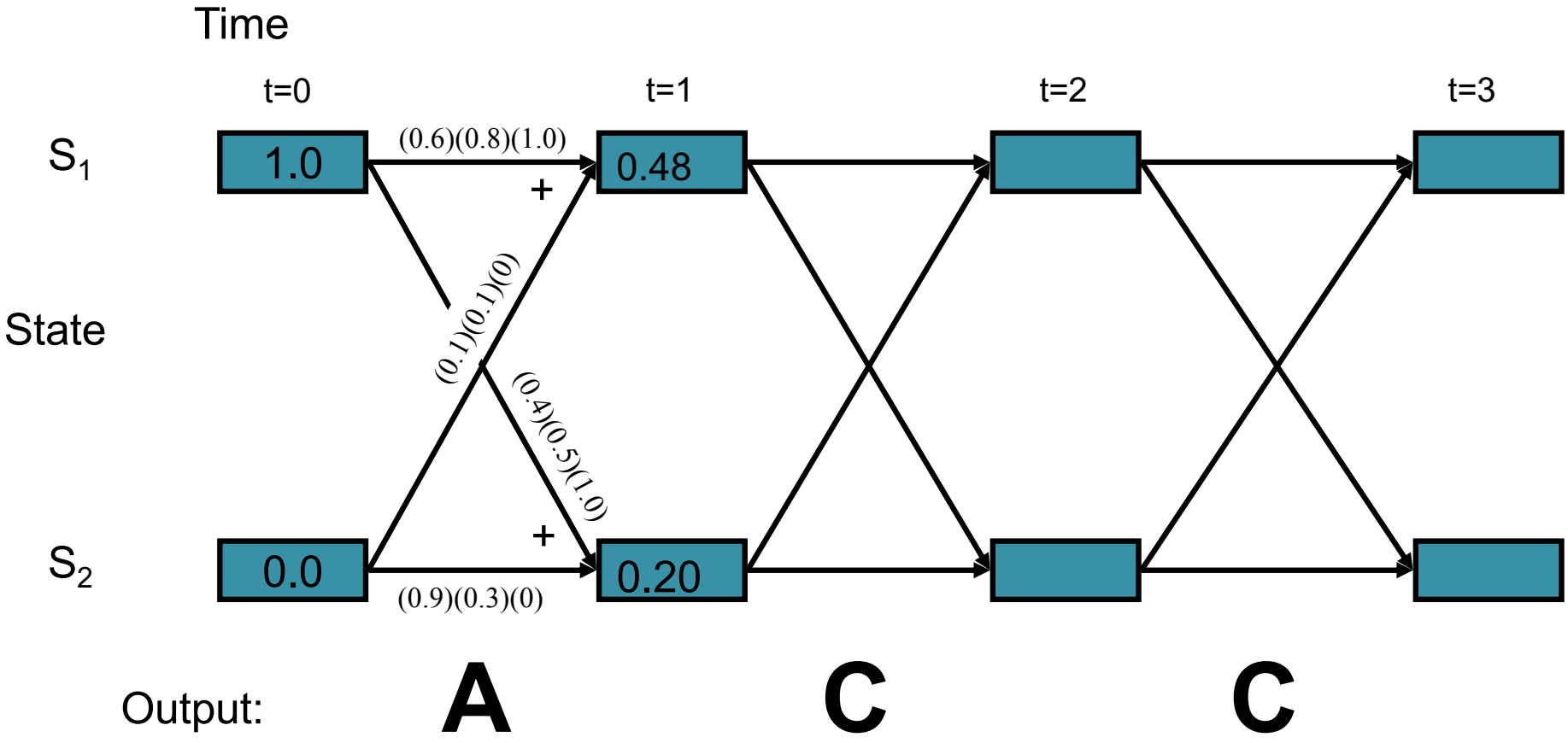
Our sample HMM



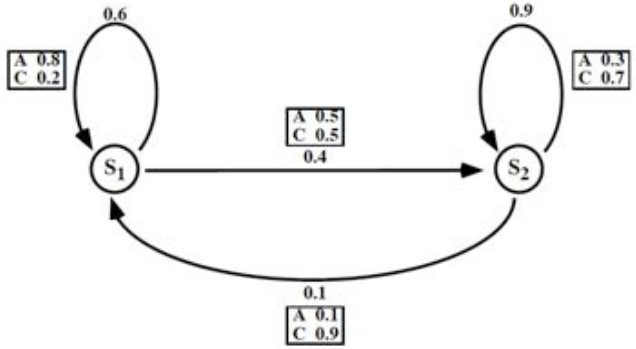
Let S_1 be initial state, S_2 be final state

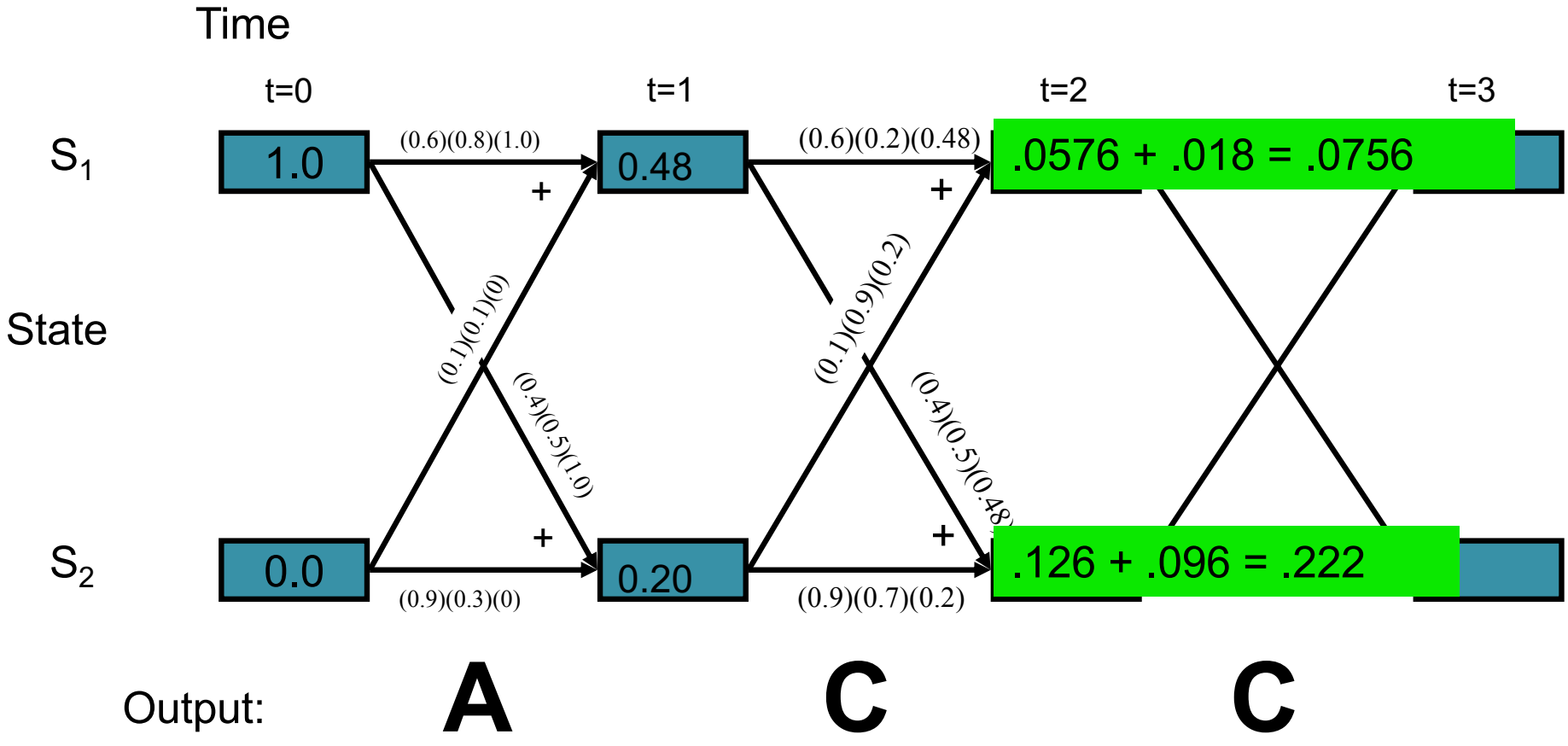
A trellis for the Forward Algorithm



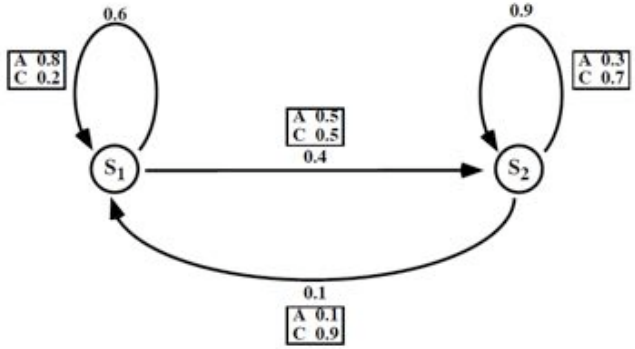


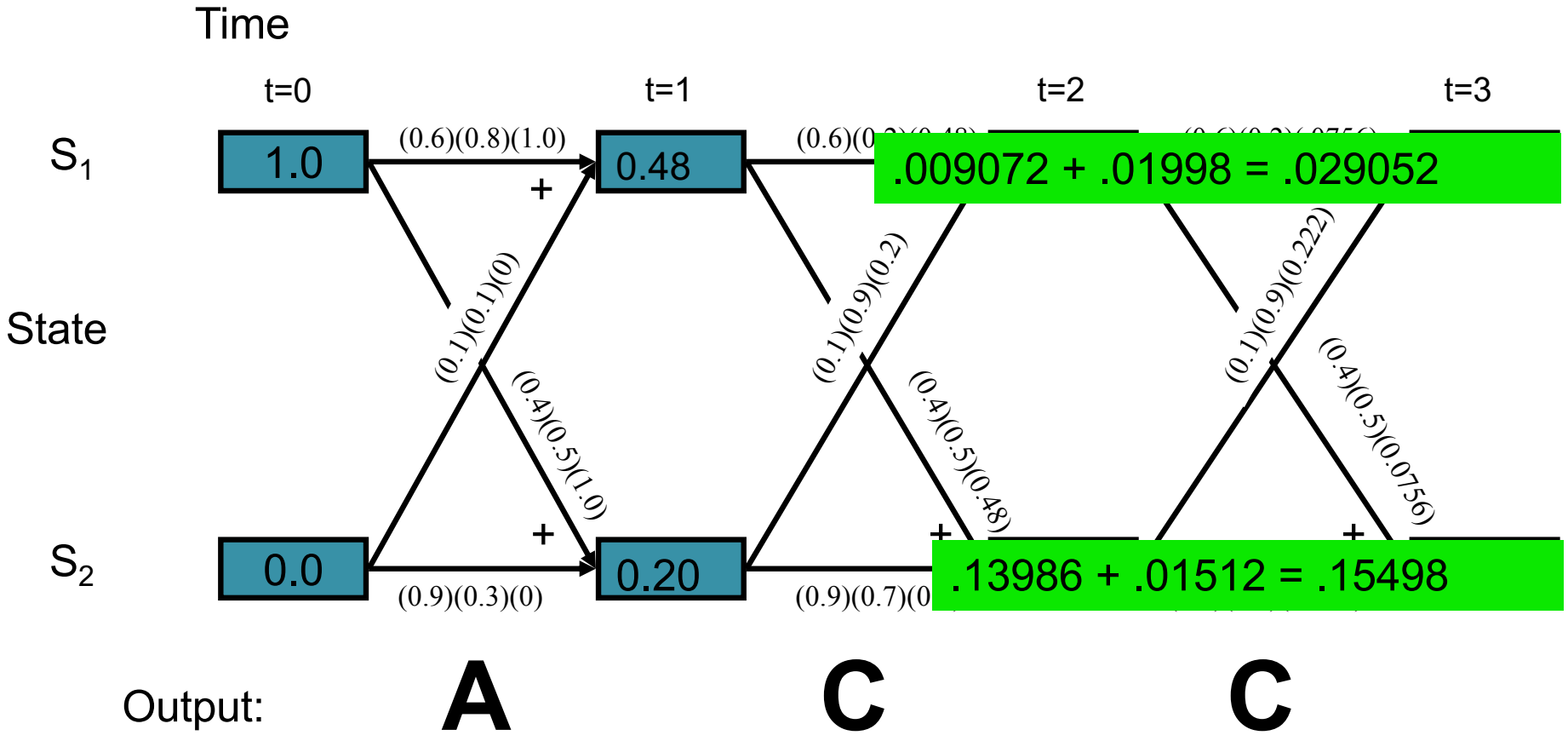
A trellis for the Forward Algorithm





A trellis for the Forward Algorithm





Probability of the model

- The Forward algorithm computes *P(y|M)*

- If we are comparing two or more models, we want the likelihood that each model generated the data: *P(M|y)*

  – Use Bayes' law: $$P(M \mid y) = \frac{P(y \mid M)P(M)}{P(y)}$$

  – Since P(y) is constant for a given input, we just need to maximize *P(y|M)P(M)*

Three classic HMM problems

2. **Decoding**: given a model and an output sequence, what is the most likely state sequence through the model that generated the output?

- A solution to this problem gives us a way to match up an observed sequence and the states in the model.

AAAGCATGCATTTAACGAGAGCACAAGGGCTCTAATGCCG

The sequence of states is an annotation of the generated string – each nucleotide is generated in intergenic, start/stop, coding state

Three classic HMM problems

2. **Decoding**: given a model and an output sequence, what is the most likely state sequence through the model that generated the output?

- A solution to this problem gives us a way to match up an observed sequence and the states in the model.

AAAGC ATG CAT TTA ACG AGA GCA CAA GGG CTC TAA TGCCG

The sequence of states is an annotation of the generated string – each nucleotide is generated in intergenic, start/stop, coding state

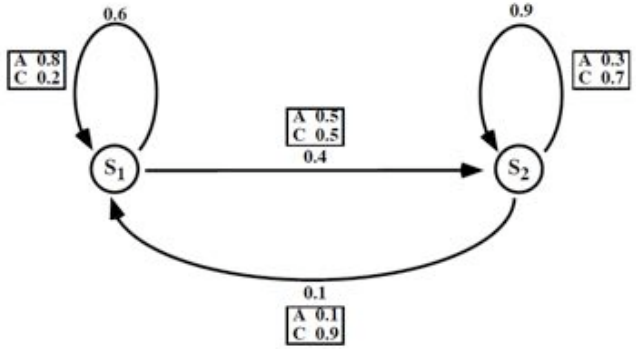
Solving the Decoding Problem: The Viterbi algorithm

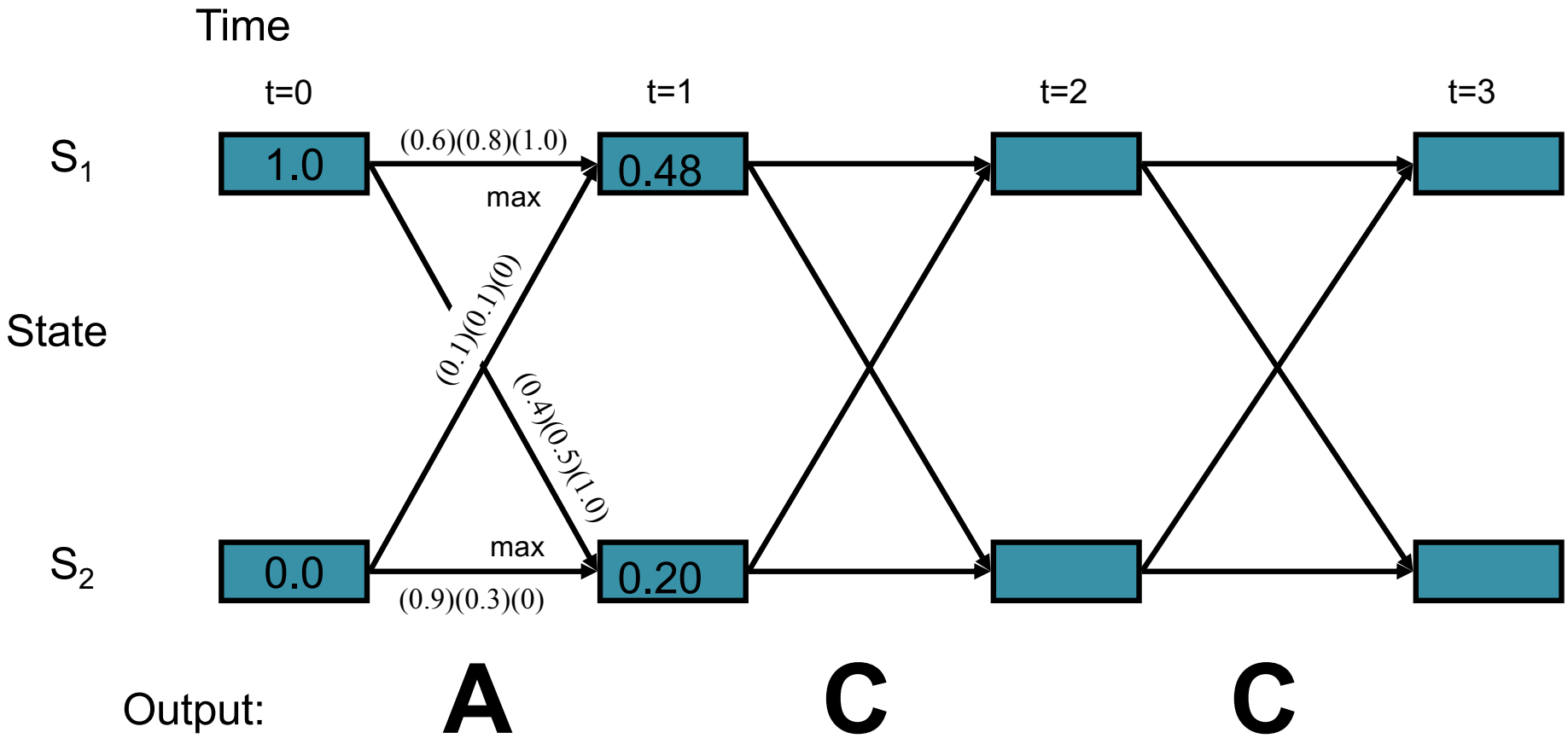
- To solve the decoding problem (find the most likely sequence of states), we evaluate the Viterbi algorithm

$$V_i(t) = \begin{cases} 0 & : \quad t = 0 \wedge i \neq S_I \\ 1 & : \quad t = 0 \wedge i = S_I \\ \max V_j(t-1) a_{ji} b_{ji}(y) & : \quad t > 0 \end{cases}$$

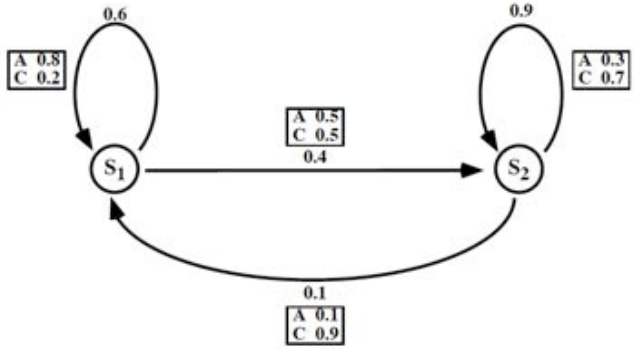
Where $V_i(t)$ is the probability that the HMM is in state i after generating the sequence $y_1, y_2, \ldots, y_t$, following the *most probable path* in the HMM

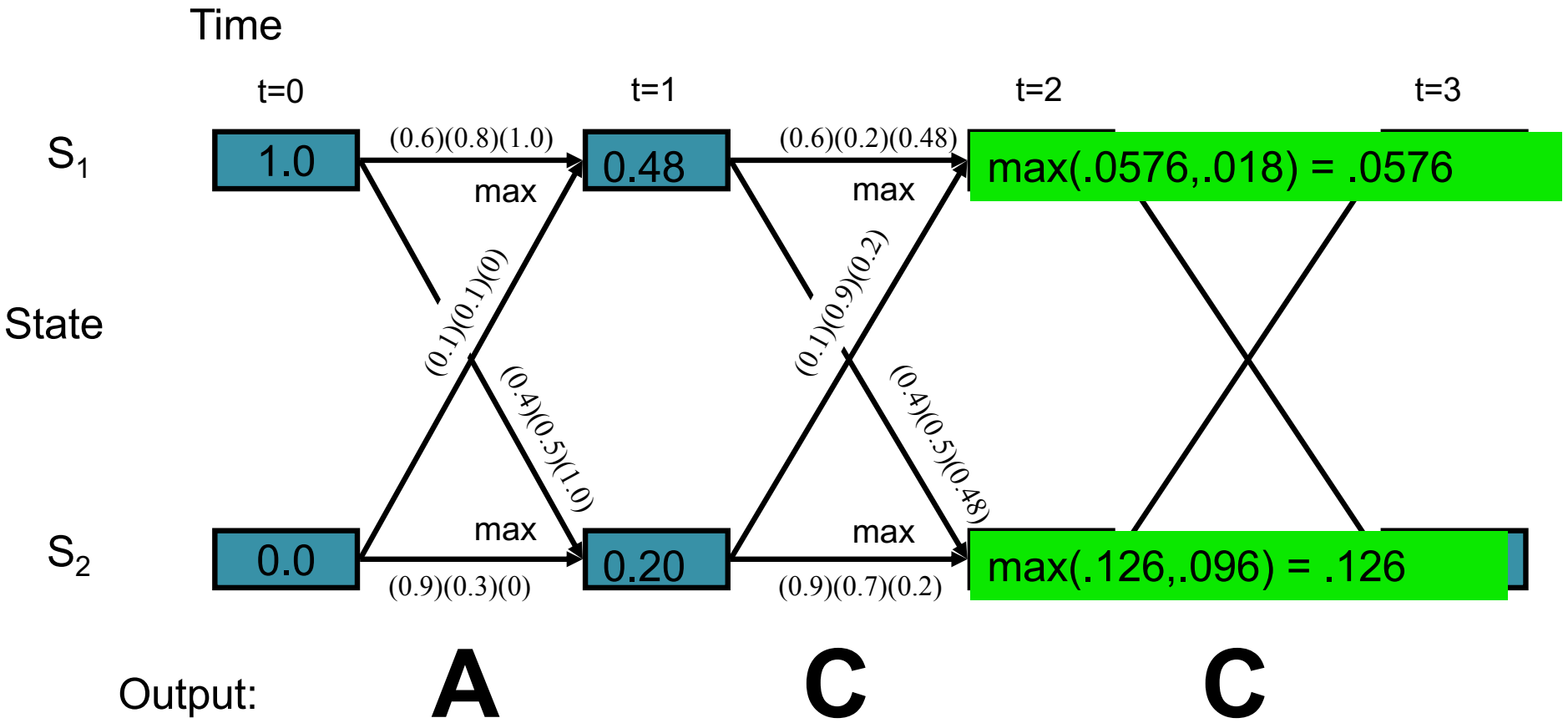
A trellis for the Viterbi Algorithm



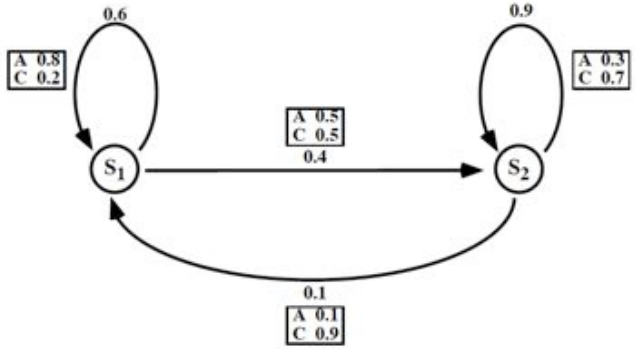


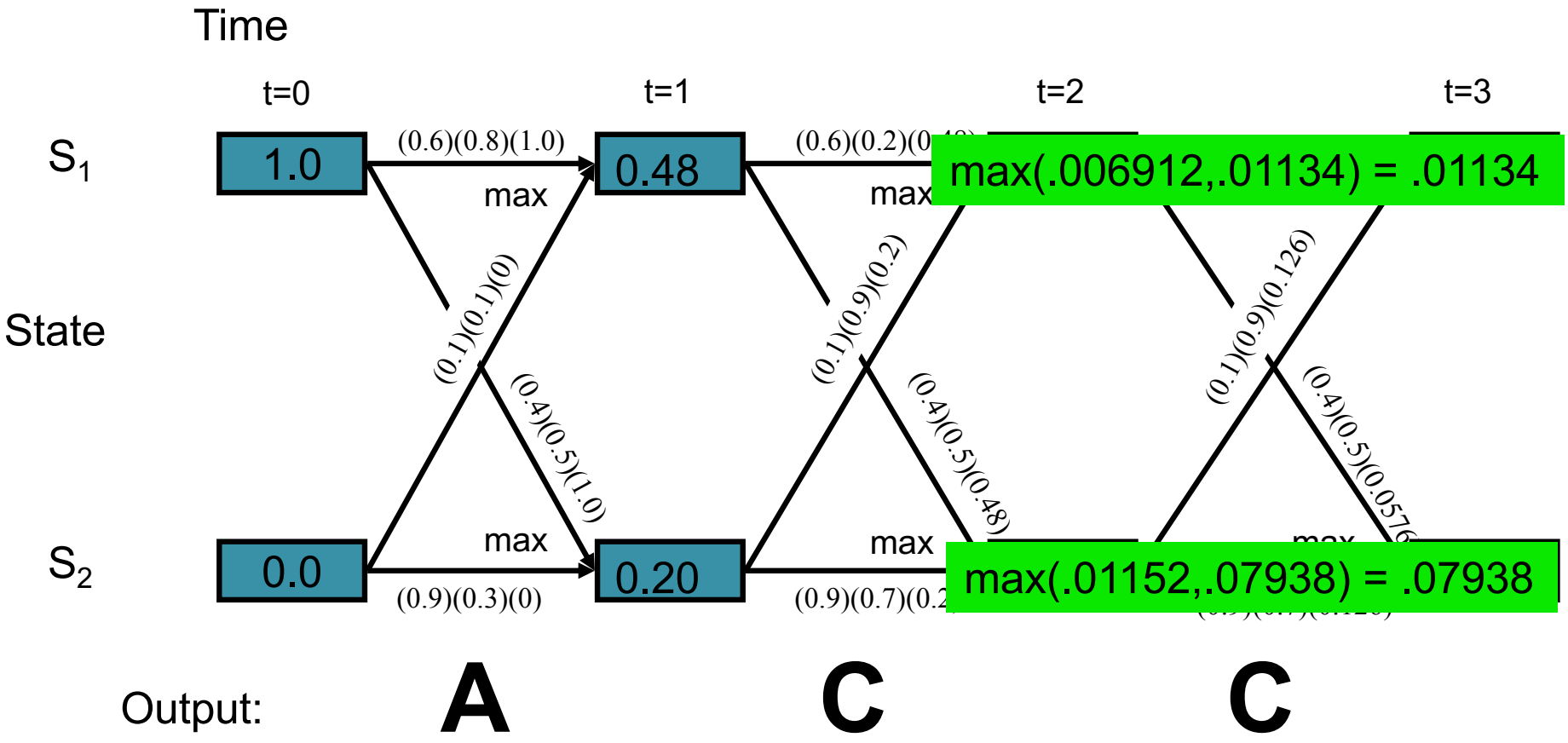
A trellis for the Viterbi Algorithm



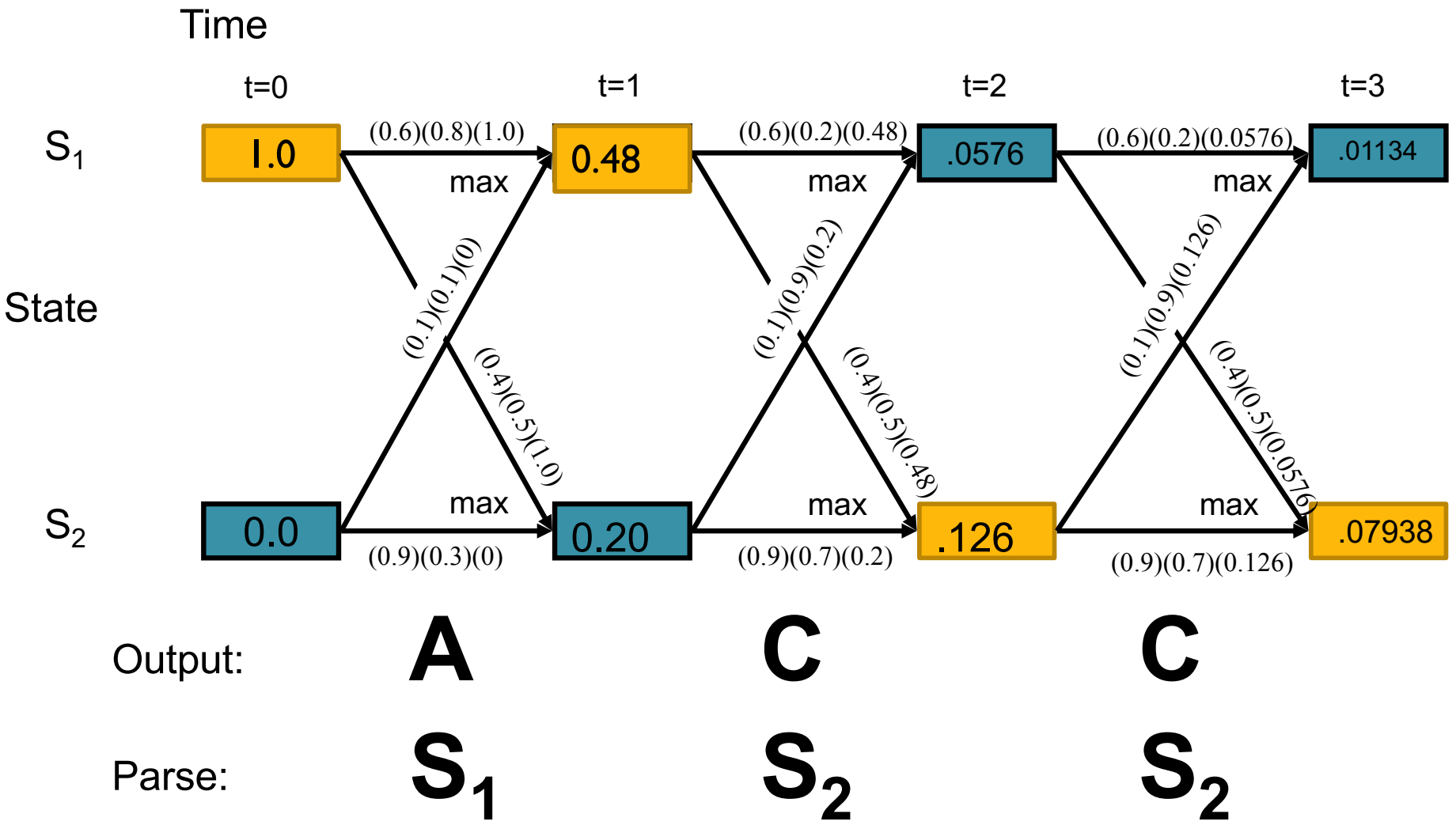


A trellis for the Viterbi Algorithm





A trellis for the Viterbi Algorithm



Three classic HMM problems

3. **Learning**: given a model and a set of observed sequences, how do we set the model's parameters so that it has a high probability of generating those sequences?

- This is perhaps the most important, and most difficult problem.

- A solution to this problem allows us to determine all the probabilities in an HMMs by using an ensemble of training data

Learning in HMMs: The E-M algorithm

- The learning algorithm is called "Expectation-Maximization" or E-M
  - Also called the Forward-Backward algorithm
  - Also called the Baum-Welch algorithm

- In order to learn the parameters in an "empty" HMM, we need:
  - The topology of the HMM
  - Data - the more the better

➔ Great topic for QB2?

Eukaryotic Gene Finding with GlimmerHMM

Mihaela Pertea

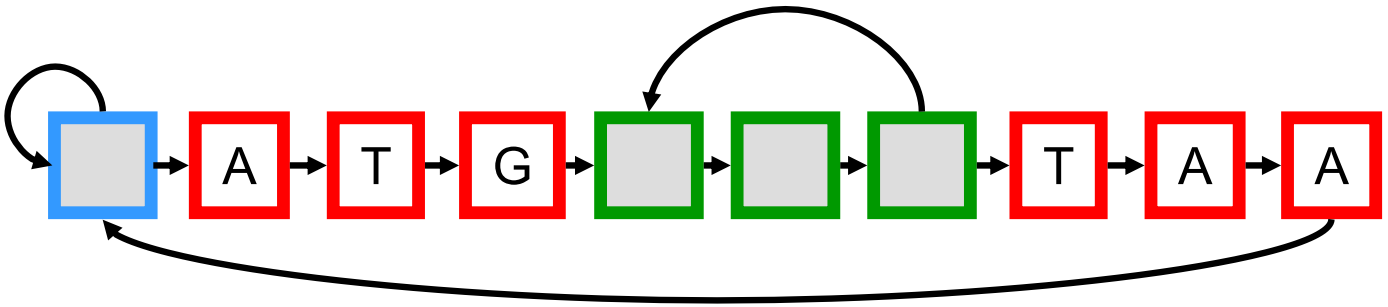
Assistant Professor

JHU

HMMs and Gene Structure

- Nucleotides {*A,C,G,T*} are the observables
- Different states generate nucleotides at different frequencies

A simple HMM for unspliced genes:



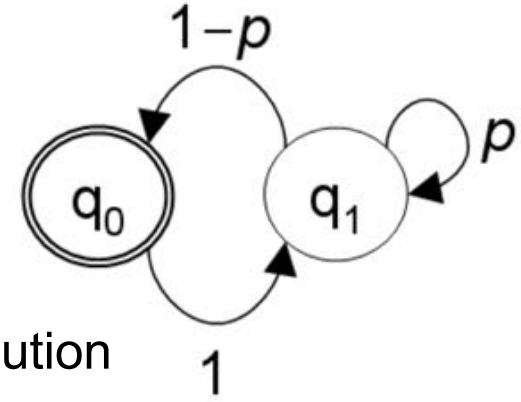
AAAGC ATG CAT TTA ACG AGA GCA CAA GGG CTC TAA TGCCG

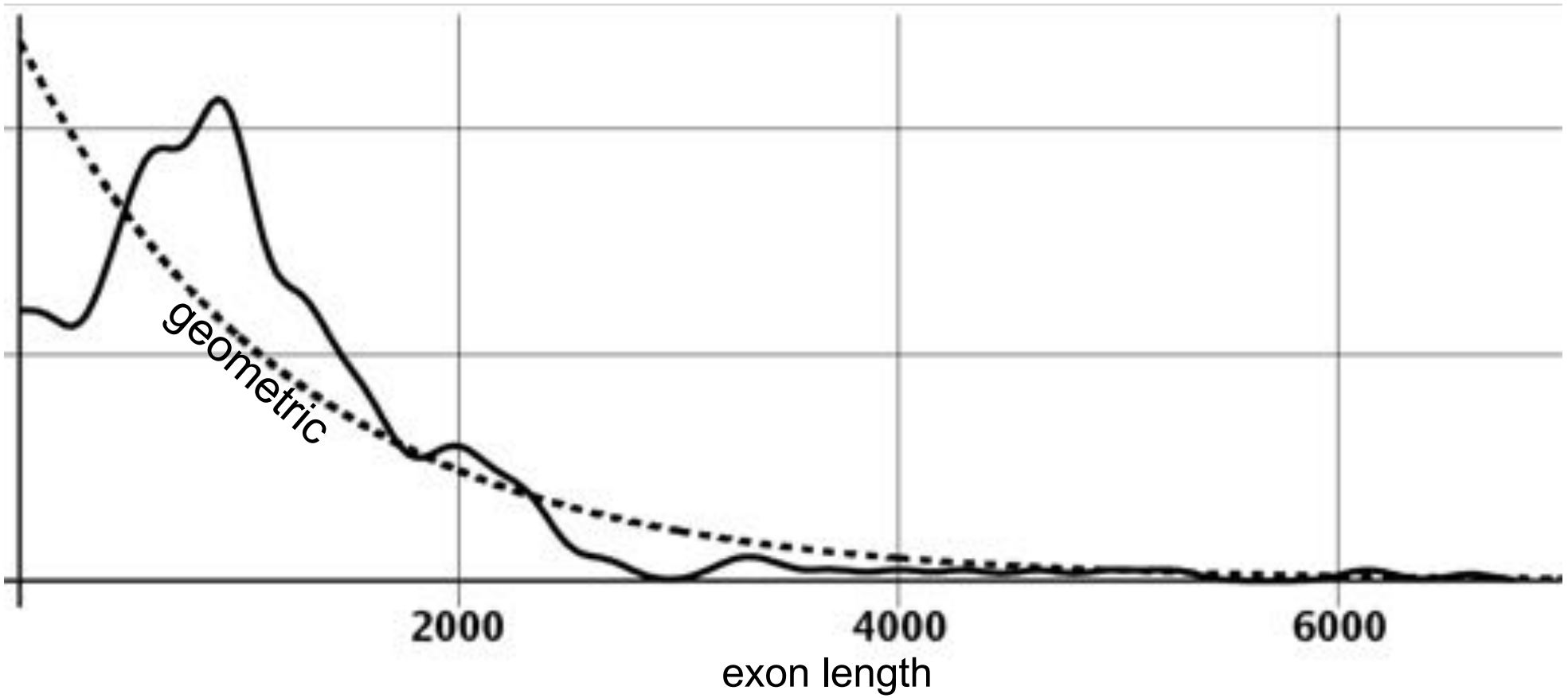
- The sequence of states is an annotation of the generated string – each nucleotide is generated in intergenic, start/stop, coding state

HMMs & Geometric Feature Lengths

$$P(x_0 \ldots x_{d-1} \mid \theta) = \left(\prod_{i=0}^{d-1} P_e(x_i \mid \theta) \right) p^{d-1}(1-p)$$

geometric distribution





Generalized HMMs Summary

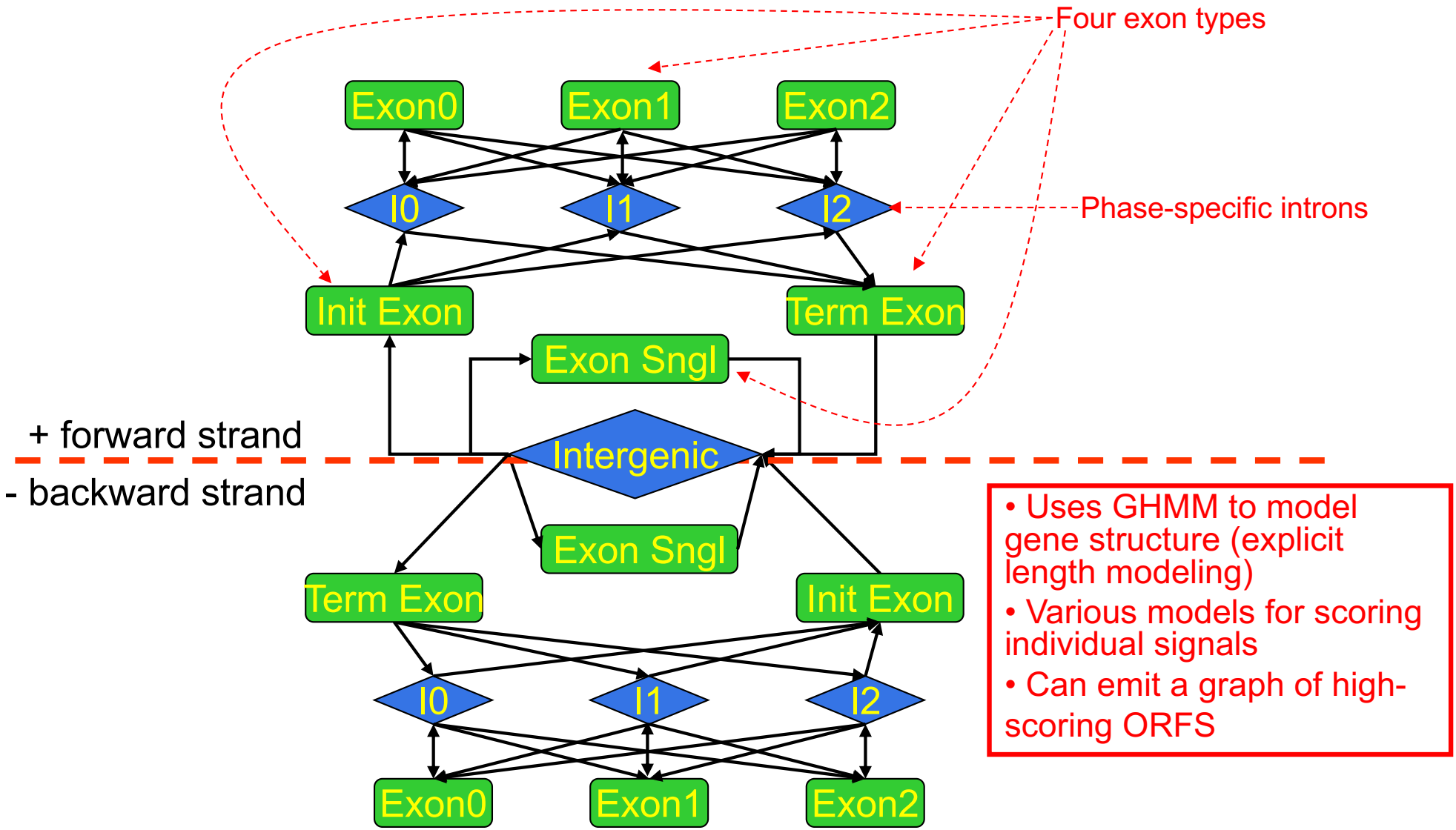
- GHMMs generalize HMMs by allowing each state to emit a subsequence rather than just a single symbol

- Whereas HMMs model all feature lengths using a geometric distribution, coding features can be modeled using an arbitrary length distribution in a GHMM

- Emission models within a GHMM can be any arbitrary probabilistic model ("submodel abstraction"), such as a neural network or decision tree

- GHMMs tend to have many fewer states => simplicity & modularity

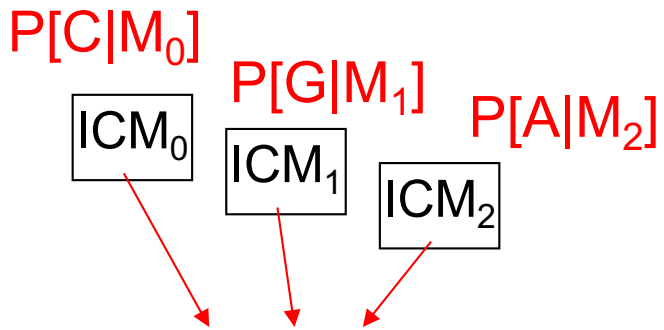
GlimmerHMM architecture



- Uses GHMM to model gene structure (explicit length modeling)
- Various models for scoring individual signals
- Can emit a graph of high-scoring ORFS

Coding vs Non-coding

A three-periodic ICM uses three ICMs in succession to evaluate the different codon positions, which have different statistics:



ATC GAT CGA TCA GCT TAT CGC ATC

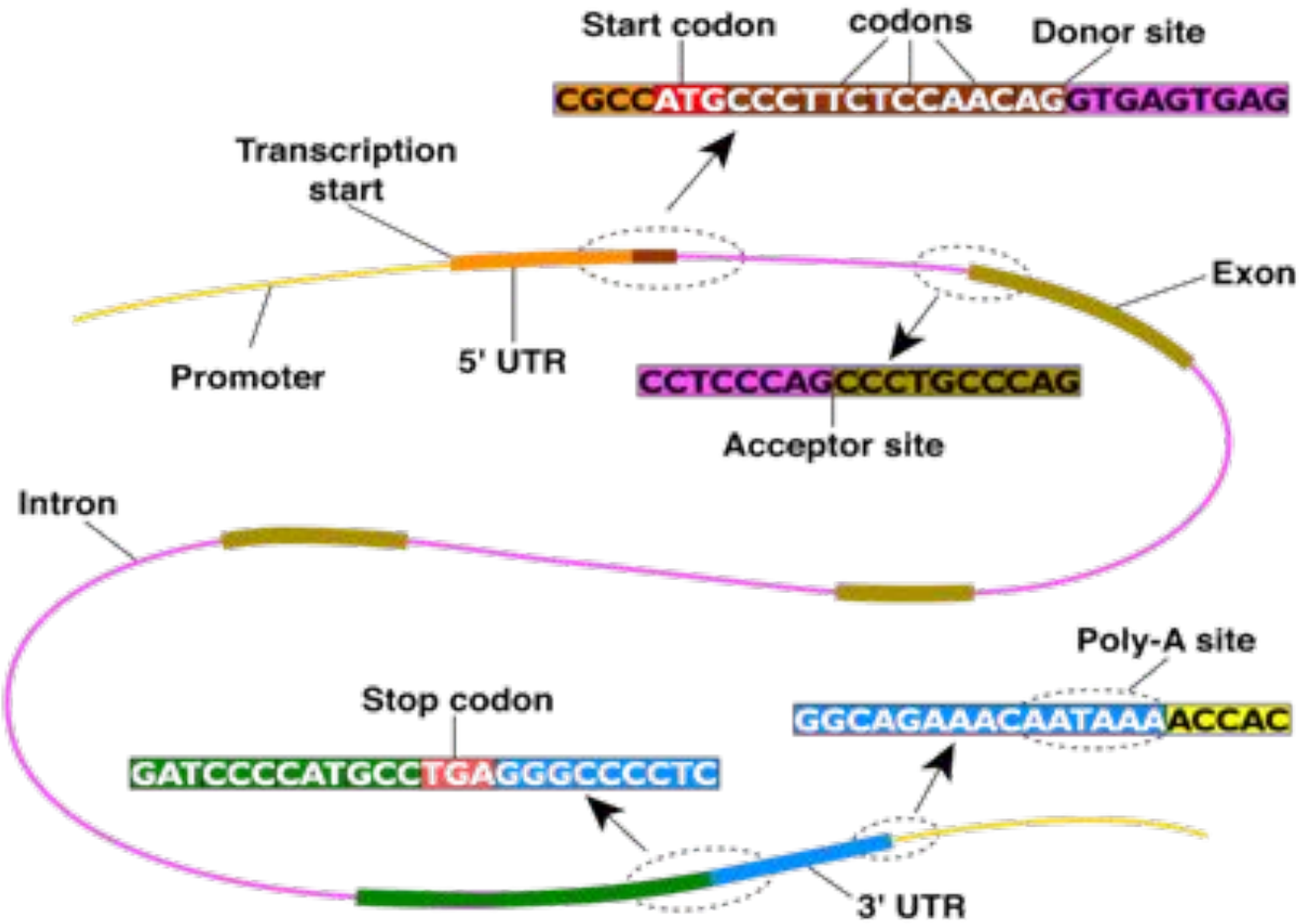
The three ICMs correspond to the three phases. Every base is evaluated in every phase, and the score for a given stretch of (putative) coding DNA is obtained by multiplying the phase-specific probabilities in a mod 3 fashion:

$$\prod_{i=0}^{L-1} P_{(f+i)(\mathrm{mod}3)}(x_i)$$

GlimmerHMM uses 3-periodic ICMs for coding and homogeneous (non-periodic) ICMs for noncoding DNA.

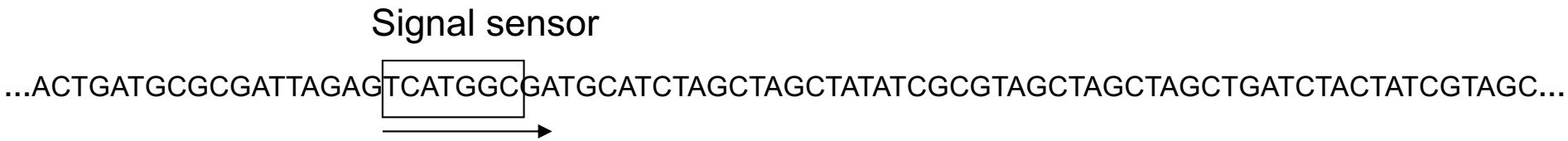
Signal Sensors

Signals – short sequence patterns in the genomic DNA that are recognized by the cellular machinery.



Identifying Signals In DNA

We slide a fixed-length model or "window" along the DNA and evaluate `score(signal)` at each point:

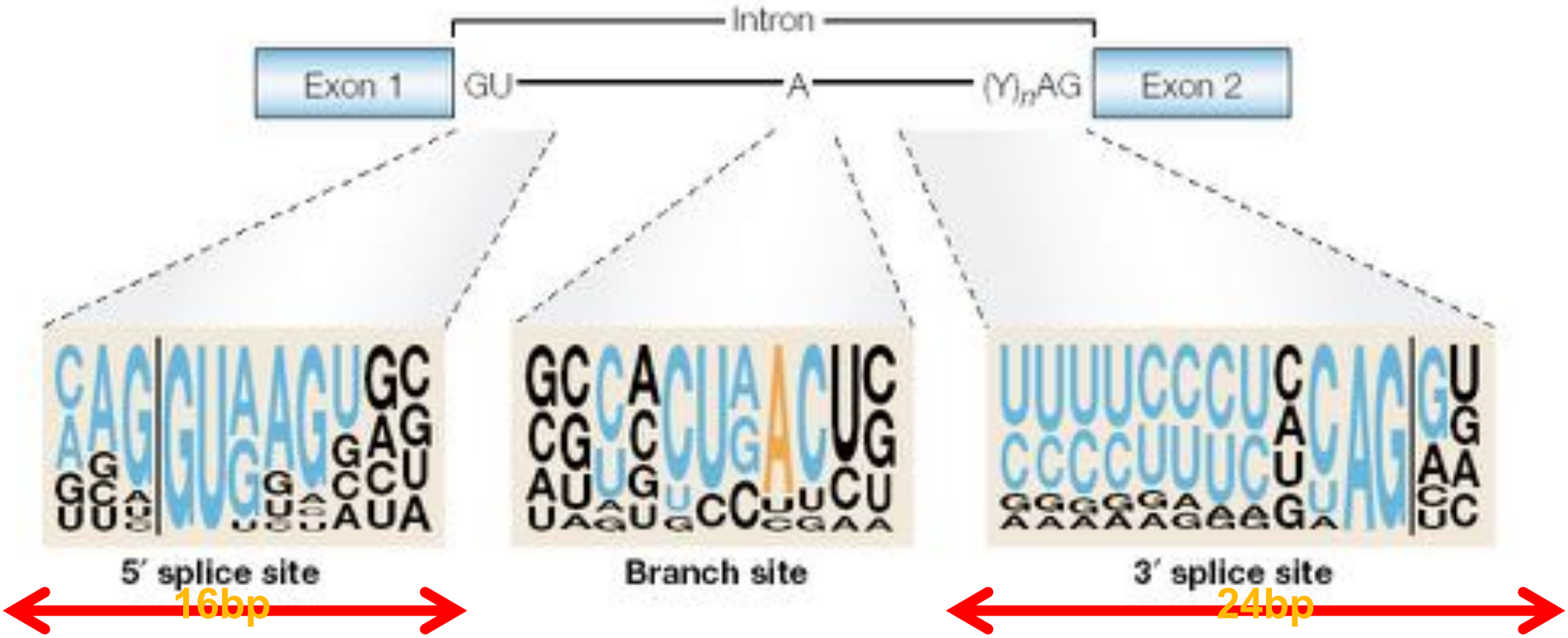


When the `score` is greater than some threshold (determined empirically to result in a desired sensitivity), we remember this position as being the potential site of a signal.

The most common signal sensor is the Weight Matrix:

A = 31% T = 28% C = 21% G = 20%	A = 18% T = 32% C = 24% G = 26%	**A** 100%	**T** 100%	**G** 100%	A = 19% T = 20% C = 29% G = 32%	A = 24% T = 18% C = 26% G = 32%

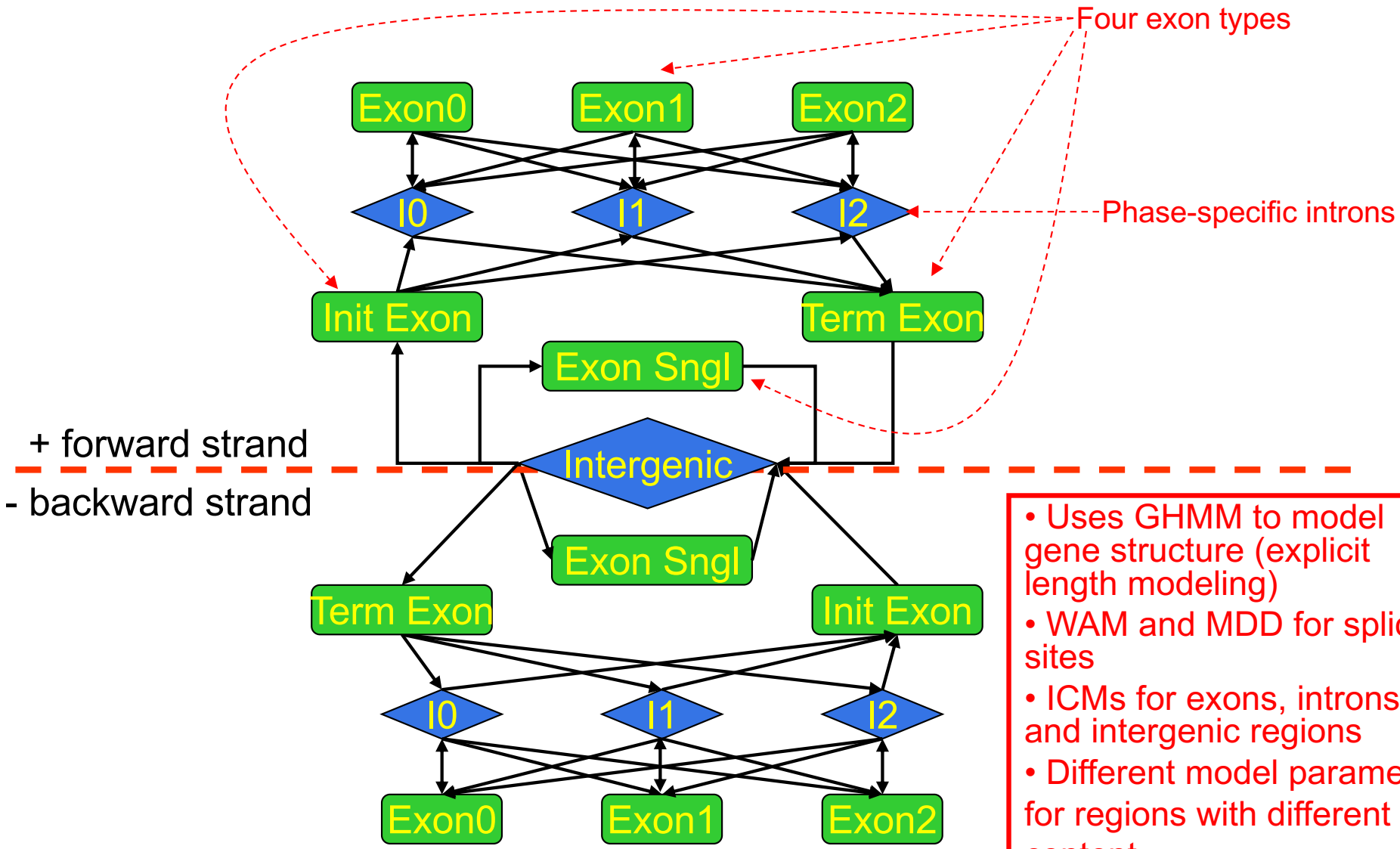
Splice site prediction



The splice site score is a combination of:

- first or second order inhomogeneous Markov models on windows around the acceptor and donor sites
- Maximal dependence decomposition (MDD) decision trees
- longer Markov models to capture difference between coding and non-coding on opposite sides of site (optional)
- maximal splice site score within 60 bp (optional)

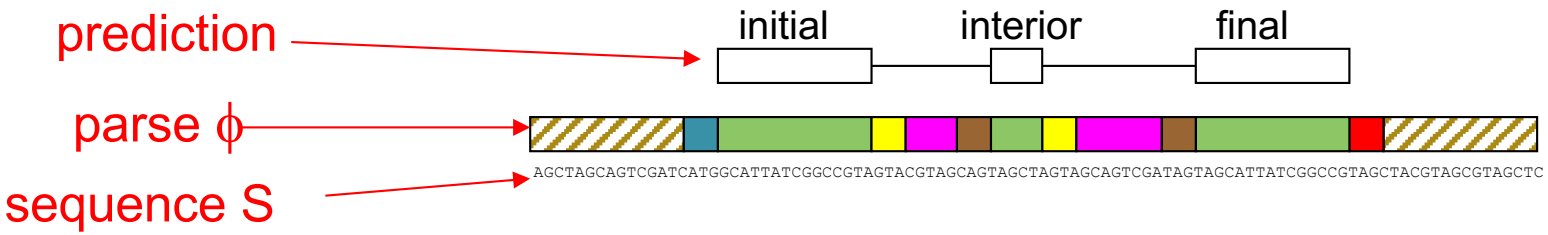
GlimmerHMM architecture



- Uses GHMM to model gene structure (explicit length modeling)
- WAM and MDD for splice sites
- ICMs for exons, introns and intergenic regions
- Different model parameters for regions with different GC content
- Can emit a graph of high-scoring ORFS

Gene Prediction with a GHMM

Given a sequence S, we would like to determine the parse ϕ of that sequence which segments the DNA into the most likely exon/intron structure:

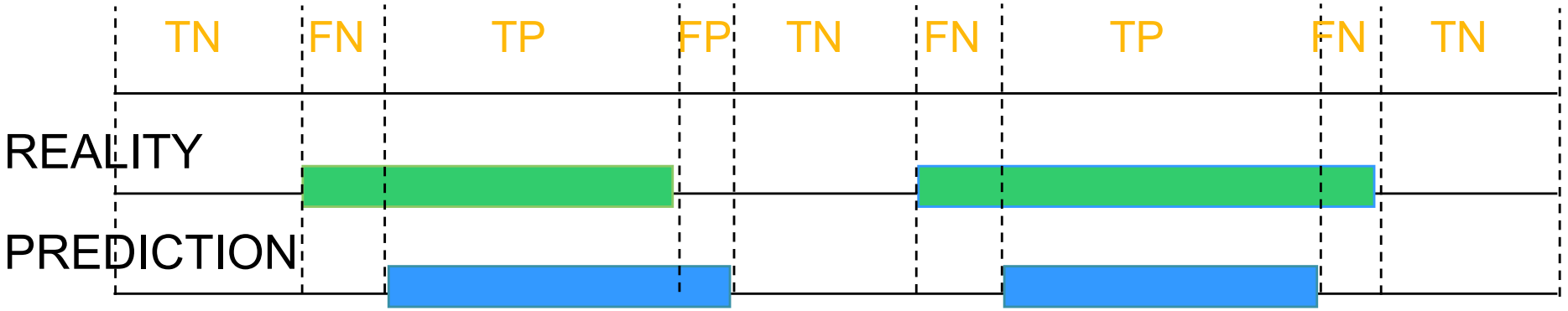


The parse ϕ consists of the coordinates of the predicted exons, and corresponds to the precise sequence of states during the operation of the GHMM (and their duration, which equals the number of symbols each state emits).

This is the same as in an HMM except that in the HMM each state emits bases with fixed probability, whereas in the GHMM each state emits an entire feature such as an exon or intron.

Evaluation of Gene Finding Programs

Nucleotide level accuracy

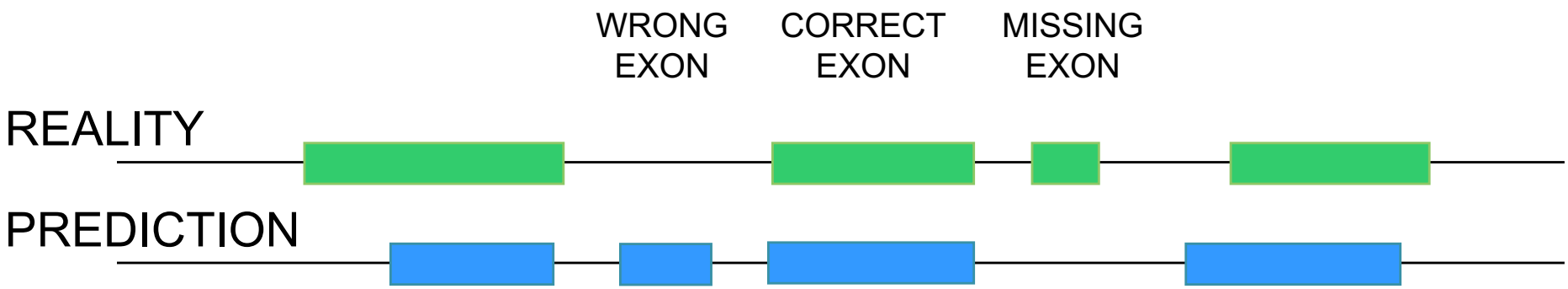


Sensitivity: $Sn = \frac{TP}{TP + FN}$ What fraction of reality did you predict?

Specificity: $Sp = \frac{TP}{TP + FP}$ What fraction of your predictions are real?

More Measures of Prediction Accuracy

Exon level accuracy



$$ExonSn = \frac{TE}{AE} = \frac{\text{number of correct exons}}{\text{number of actual exons}}$$

$$ExonSp = \frac{TE}{PE} = \frac{\text{number of correct exons}}{\text{number of predicted exons}}$$

GlimmerHMM is a high-performance ab initio gene finder

Arabidopsis thaliana test results

	Nucleotide			Exon			Gene		
	Sn	Sp	**Acc**	Sn	Sp	**Acc**	Sn	Sp	**Acc**
GlimmerHMM	97	99	**98**	84	89	**86.5**	60	61	**60.5**
SNAP	96	99	**97.5**	83	85	**84**	60	57	**58.5**
Genscan+	93	99	**96**	74	81	**77.5**	35	35	**35**

- All three programs were tested on a test data set of 809 genes, which did not overlap with the training data set of GlimmerHMM.
- All genes were confirmed by full-length Arabidopsis cDNAs and carefully inspected to remove homologues.

GlimmerHMM on human data

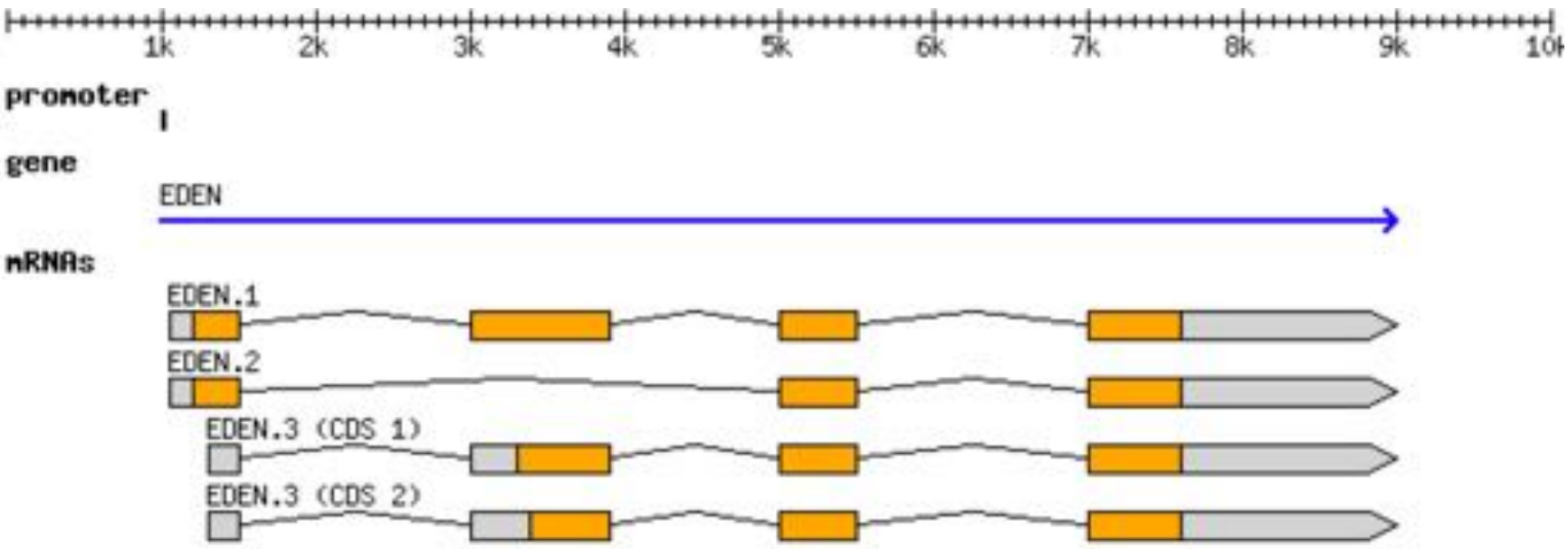
	Nuc Sens	*Nuc Spec*	*Nuc Acc*	*Exon Sens*	*Exon Spec*	*Exon Acc*	*Exact Genes*
GlimmerHMM	86%	72%	79%	72%	62%	67%	17%
Genscan	86%	68%	77%	69%	60%	65%	13%

GlimmerHMM's performace compared to Genscan on 963 human RefSeq genes selected randomly from all 24 chromosomes, non-overlapping with the training set. The test set contains 1000 bp of untranslated sequence on either side (5' or 3') of the coding portion of each gene.

Gene Finding Overview

- Prokaryotic gene finding distinguishes real genes and random ORFs
  - Prokaryotic genes have simple structure and are largely homogenous, making it relatively easy to recognize their sequence composition
- Eukaryotic gene finding identifies the genome-wide most probable gene models (set of exons)
  - "Probabilistic Graphical Model" to enforce overall gene structure, separate models to score splicing/transcription signals
  - Accuracy depends to a large extent on the quality of the training data

Gene Models



- "Generic Feature Format" (GFF) records genomic features
  - Coordinates of each exon
  - Coordinates of UTRs
  - Link together exons into transcripts
  - Link together transcripts into gene models

http://www.sequenceontology.org/gff3.shtml

GFF File format

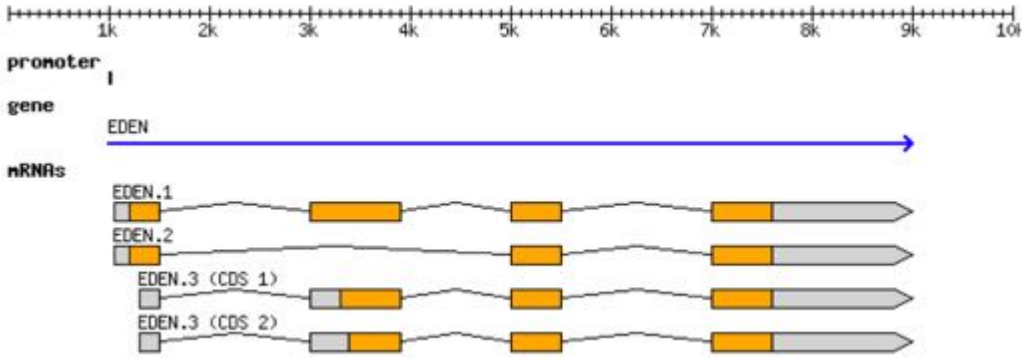
GFF3 files are nine-column, tab-delimited, plain text files

1. ***seqid:*** The ID of the sequence
2. ***source:*** Algorithm or database that generated this feature
3. ***type:*** *gene/exon/CDS/etc…*
4. ***start:*** 1-based coordinate
5. ***end***: 1-based coordinate
6. ***score***: E-values/p-values/index/colors/…
7. ***strand:*** "+' for positive "-" for minus, "." not stranded
8. ***phase:*** For "CDS", where the feature begins with reference to the reading frame (0,1,2)
9. ***attributes:*** A list of tag=value features

   Parent: Indicates the parent of the feature (group exons into transcripts, transcripts into genes, …)

GFF Example

Gene "EDEN" with 3 alternatively spliced transcripts, isoform 3 has two alternative translation start sites



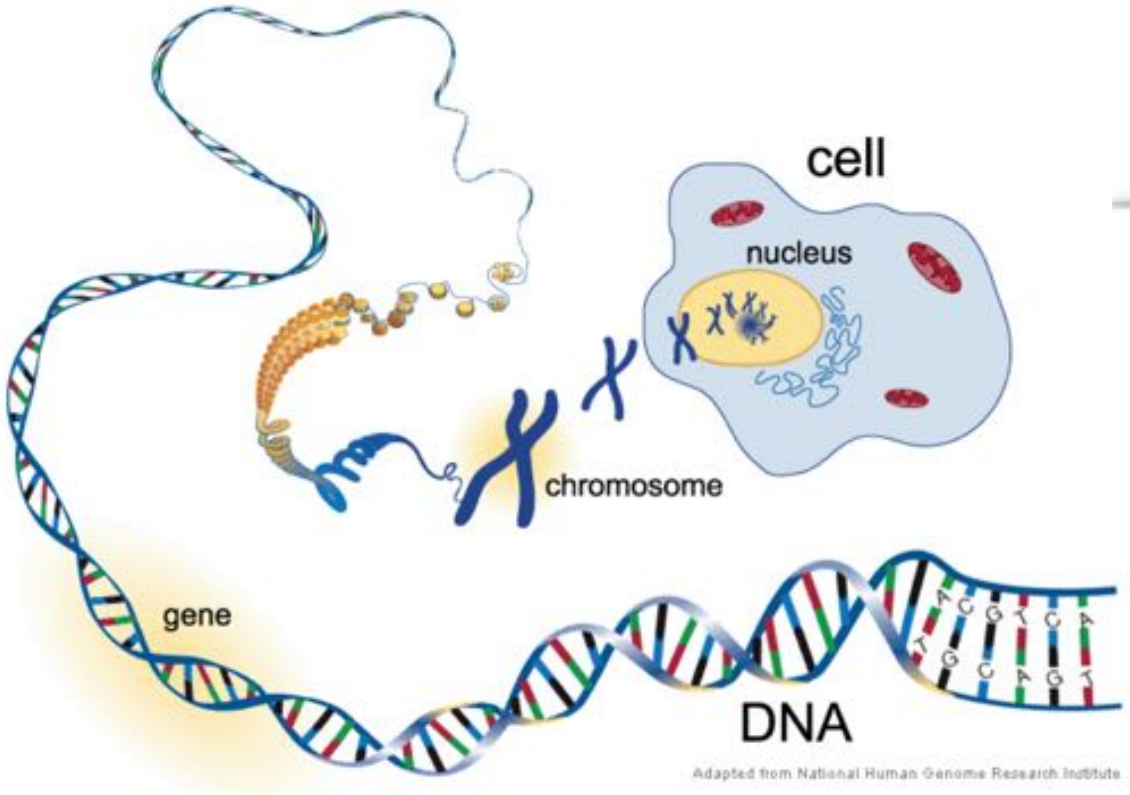

```
##gff-version 3
##sequence-region   ctg123 1 1497228
ctg123 . gene            1000  9000  .  +  .  ID=gene00001;Name=EDEN

ctg123 . TF_binding_site 1000  1012  .  +  .  ID=tfbs00001;Parent=gene00001

ctg123 . mRNA            1050  9000  .  +  .  ID=mRNA00001;Parent=gene00001;Name=EDEN.1
ctg123 . mRNA            1050  9000  .  +  .  ID=mRNA00002;Parent=gene00001;Name=EDEN.2
ctg123 . mRNA            1300  9000  .  +  .  ID=mRNA00003;Parent=gene00001;Name=EDEN.3

ctg123 . exon            1300  1500  .  +  .  ID=exon00001;Parent=mRNA00003
ctg123 . exon            1050  1500  .  +  .  ID=exon00002;Parent=mRNA00001,mRNA00002
ctg123 . exon            3000  3902  .  +  .  ID=exon00003;Parent=mRNA00001,mRNA00003
ctg123 . exon            5000  5500  .  +  .  ID=exon00004;Parent=mRNA00001,mRNA00002,mRNA00003
ctg123 . exon            7000  9000  .  +  .  ID=exon00005;Parent=mRNA00001,mRNA00002,mRNA00003

ctg123 . CDS             1201  1500  .  +  0  ID=cds00001;Parent=mRNA00001;Name=edenprotein.1
ctg123 . CDS             3000  3902  .  +  0  ID=cds00001;Parent=mRNA00001;Name=edenprotein.1
ctg123 . CDS             5000  5500  .  +  0  ID=cds00001;Parent=mRNA00001;Name=edenprotein.1
ctg123 . CDS             7000  7600  .  +  0  ID=cds00001;Parent=mRNA00001;Name=edenprotein.1

ctg123 . CDS             1201  1500  .  +  0  ID=cds00002;Parent=mRNA00002;Name=edenprotein.2
ctg123 . CDS             5000  5500  .  +  0  ID=cds00002;Parent=mRNA00002;Name=edenprotein.2
ctg123 . CDS             7000  7600  .  +  0  ID=cds00002;Parent=mRNA00002;Name=edenprotein.2

ctg123 . CDS             3301  3902  .  +  0  ID=cds00003;Parent=mRNA00003;Name=edenprotein.3
ctg123 . CDS             5000  5500  .  +  1  ID=cds00003;Parent=mRNA00003;Name=edenprotein.3
ctg123 . CDS             7000  7600  .  +  1  ID=cds00003;Parent=mRNA00003;Name=edenprotein.3

ctg123 . CDS             3391  3902  .  +  0  ID=cds00004;Parent=mRNA00003;Name=edenprotein.4
ctg123 . CDS             5000  5500  .  +  1  ID=cds00004;Parent=mRNA00003;Name=edenprotein.4
ctg123 . CDS             7000  7600  .  +  1  ID=cds00004;Parent=mRNA00003;Name=edenprotein.4
```

Outline

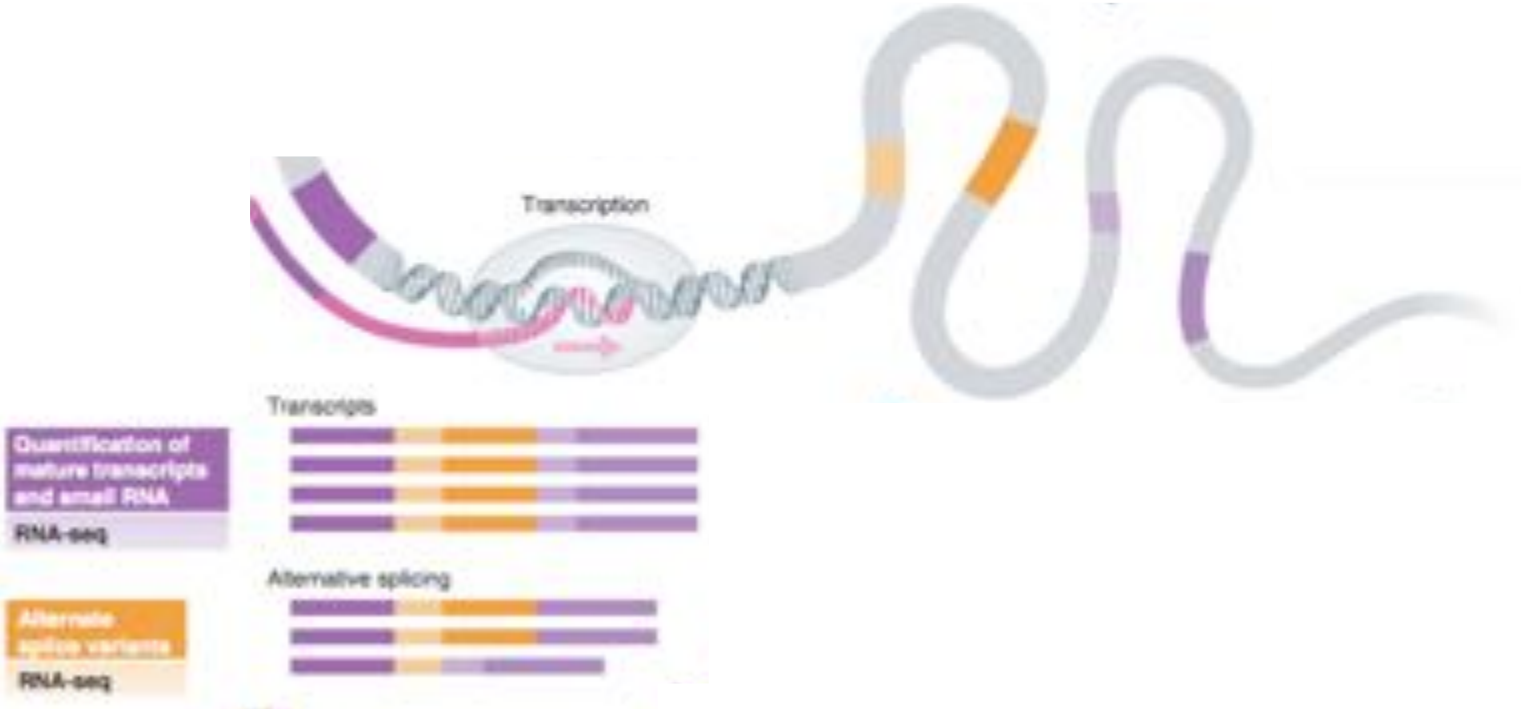
1. Alignment to other genomes
2. Prediction aka "Gene Finding"
3. Experimental & Functional Assays

Sequencing techniques

Much of the capacity is used to sequence genomes (or exomes) of individuals…



… but biology is much more than just genomes…



Soon et al., Molecular Systems Biology, 2013

Sequencing Assays

The *Seq List (in chronological order)

1. Gregory E. Crawford et al., "Genome-wide Mapping of DNase Hypersensitive Sites Using Massively Parallel Signature Sequencing (MPSS)," Genome Research 16, no. 1 (January 1, 2006): 123–131, doi:10.1101/gr.4074106.
2. David S. Johnson et al., "Genome-Wide Mapping of in Vivo Protein-DNA Interactions," Science 316, no. 5830 (June 8, 2007): 1497–1502, doi:10.1126/science.1141319.
3. Tarjei S. Mikkelsen et al., "Genome-wide Maps of Chromatin State in Pluripotent and Lineage-committed Cells," Nature 448, no. 7153 (August 2, 2007): 553–560, doi:10.1038/nature06008.
4. Thomas A. Down et al., "A Bayesian Deconvolution Strategy for Immunoprecipitation-based DNA Methylome Analysis," Nature Biotechnology 26, no. 7 (July 2008): 779–785, doi:10.1038/nbt1414.
5. Ali Mortazavi et al., "Mapping and Quantifying Mammalian Transcriptomes by RNA-Seq," Nature Methods 5, no. 7 (July 2008): 621–628, doi:10.1038/nmeth.1226.
6. Nathan A. Baird et al., "Rapid SNP Discovery and Genetic Mapping Using Sequenced RAD Markers," PLoS ONE 3, no. 10 (October 13, 2008): e3376, doi:10.1371/journal.pone.0003376.
7. Leighton J. Core, Joshua J. Waterfall, and John T. Lis, "Nascent RNA Sequencing Reveals Widespread Pausing and Divergent Initiation at Human Promoters," Science 322, no. 5909 (December 19, 2008): 1845–1848, doi:10.1126/science.1162228.
8. Chao Xie and Martti T. Tammi, "CNV-seq, a New Method to Detect Copy Number Variation Using High-throughput Sequencing," BMC Bioinformatics 10, no. 1 (March 6, 2009): 80, doi:10.1186/1471-2105-10-80.
9. Jay R. Hesselberth et al., "Global Mapping of protein-DNA Interactions in Vivo by Digital Genomic Footprinting," Nature Methods 6, no. 4 (April 2009): 283–289, doi:10.1038/nmeth.1313.
10. Nicholas T. Ingolia et al., "Genome-Wide Analysis in Vivo of Translation with Nucleotide Resolution Using Ribosome Profiling," Science 324, no. 5924 (April 10, 2009): 218–223, doi:10.1126/science.1168978.
11. Alayne L. Brunner et al., "Distinct DNA Methylation Patterns Characterize Differentiated Human Embryonic Stem Cells and Developing Human Fetal Liver," Genome Research 19, no. 6 (June 1, 2009): 1044–1056, doi:10.1101/gr.088773.108.
12. Mayumi Oda et al., "High-resolution Genome-wide Cytosine Methylation Profiling with Simultaneous Copy Number Analysis and Optimization for Limited Cell Numbers," Nucleic Acids Research 37, no. 12 (July 1, 2009): 3829–3839, doi:10.1093/nar/gkp260.
13. Zachary D. Smith et al., "High-throughput Bisulfite Sequencing in Mammalian Genomes," Methods 48, no. 3 (July 2009): 226–232, doi:10.1016/j.ymeth.2009.05.003.
14. Andrew M. Smith et al., "Quantitative Phenotyping via Deep Barcode Sequencing," Genome Research (July 21, 2009),

What is a *Seq assay?

Molecular Biology

Mathematics & Computer Science

Computational Biology

Desired measurement → **reduce to sequencing** → Sequence → Solve inverse problem → Analyze

Creativity

Chemistry & Physics

Statistics

Biology

$$\mathbb{P}(f=(p,t,l)) \approx \frac{\lambda_l \cdot \frac{\tau_t}{\bar{l}(t)} \cdot w_{p|t,l} \cdot \phi_{f|p,t,l}}{\sum_{(q,r,m)\in\mathcal{A}(f)} \lambda_m \cdot \frac{\tau_r}{\bar{l}(r)} \cdot w_{q|r,m} \cdot \phi_{f|q,r,m}}$$

Annotation Summary

- Three major approaches to annotate a genome
  1. Alignment:
     - Does this sequence align to any other sequences of known function?
     - Great for projecting knowledge from one species to another
  2. Prediction:
     - Does this sequence statistically resemble other known sequences?
     - Potentially most flexible but dependent on good training data
  3. Experimental:
     - Lets test to see if it is transcribed/methylated/bound/etc
     - Strongest but expensive and context dependent

- Many great resources available
  - Learn to love the literature and the databases
  - Standard formats let you rapidly query and cross reference
  - Google is your number one resource ☺


Welcome to Applied Comparative Genomics
https://github.com/schatzlab/appliedgenomics2018

Questions?