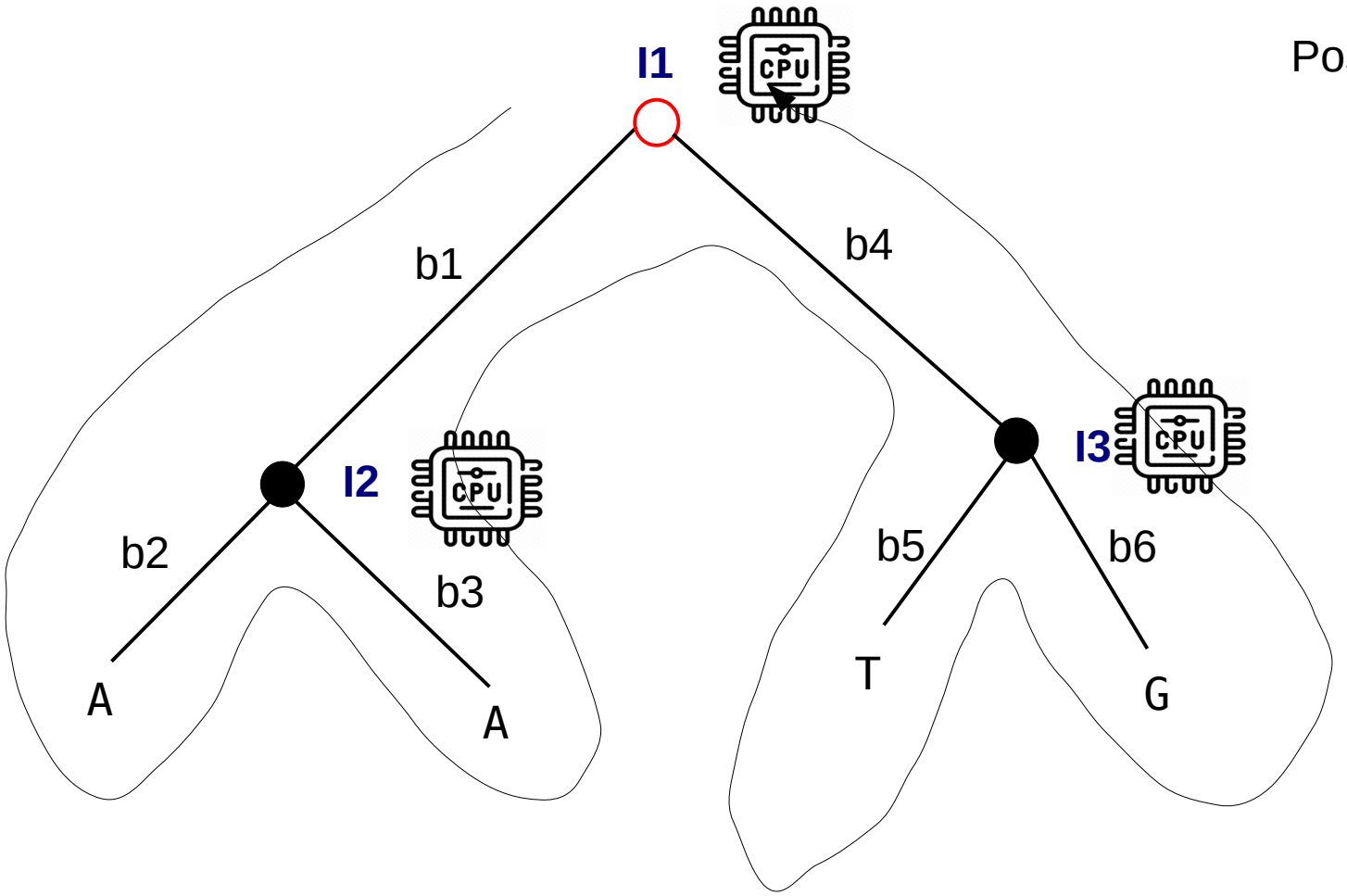
Introduction to Bioinformatics for Computer Scientists

Lecture 8

Outline

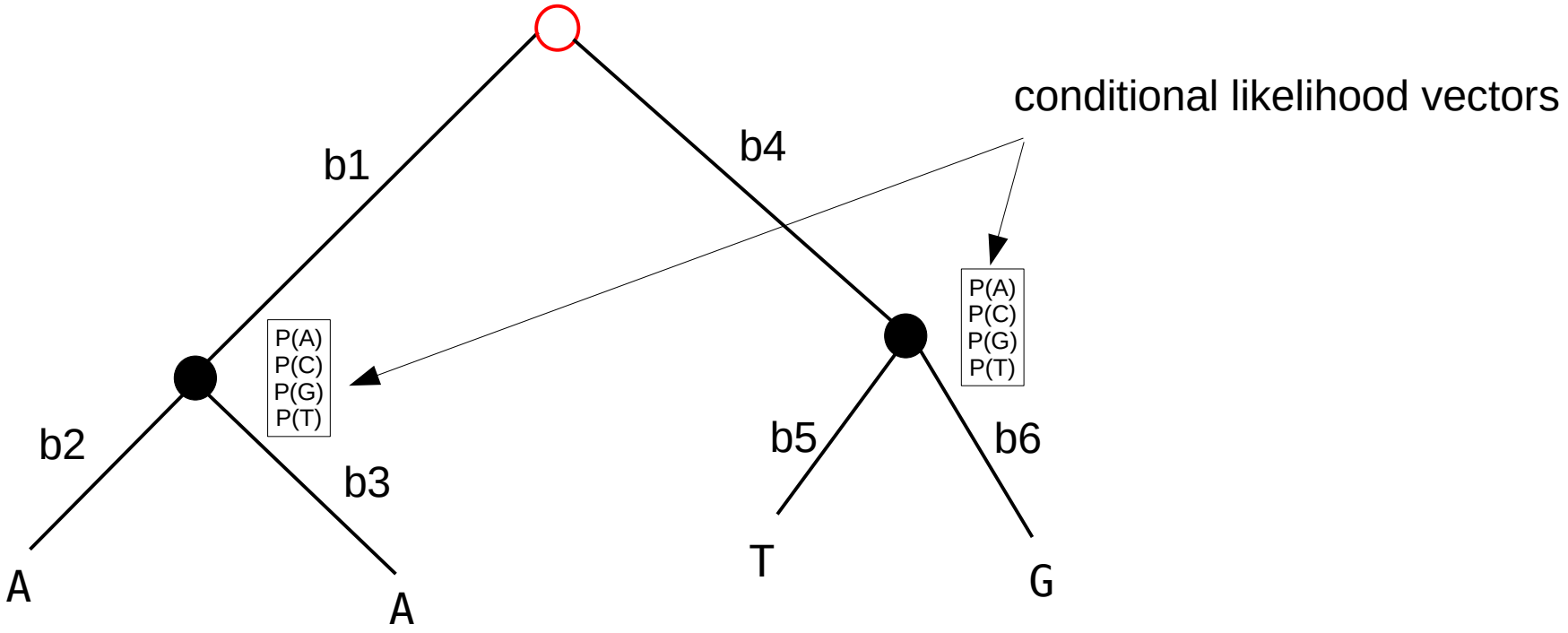
- Last time:
  - How to Compute the Likelihood of a tree
  - How to compute the Likelihood efficiently: **Felsenstein Pruning Algorithm**

The Felsenstein Pruning Algorithm

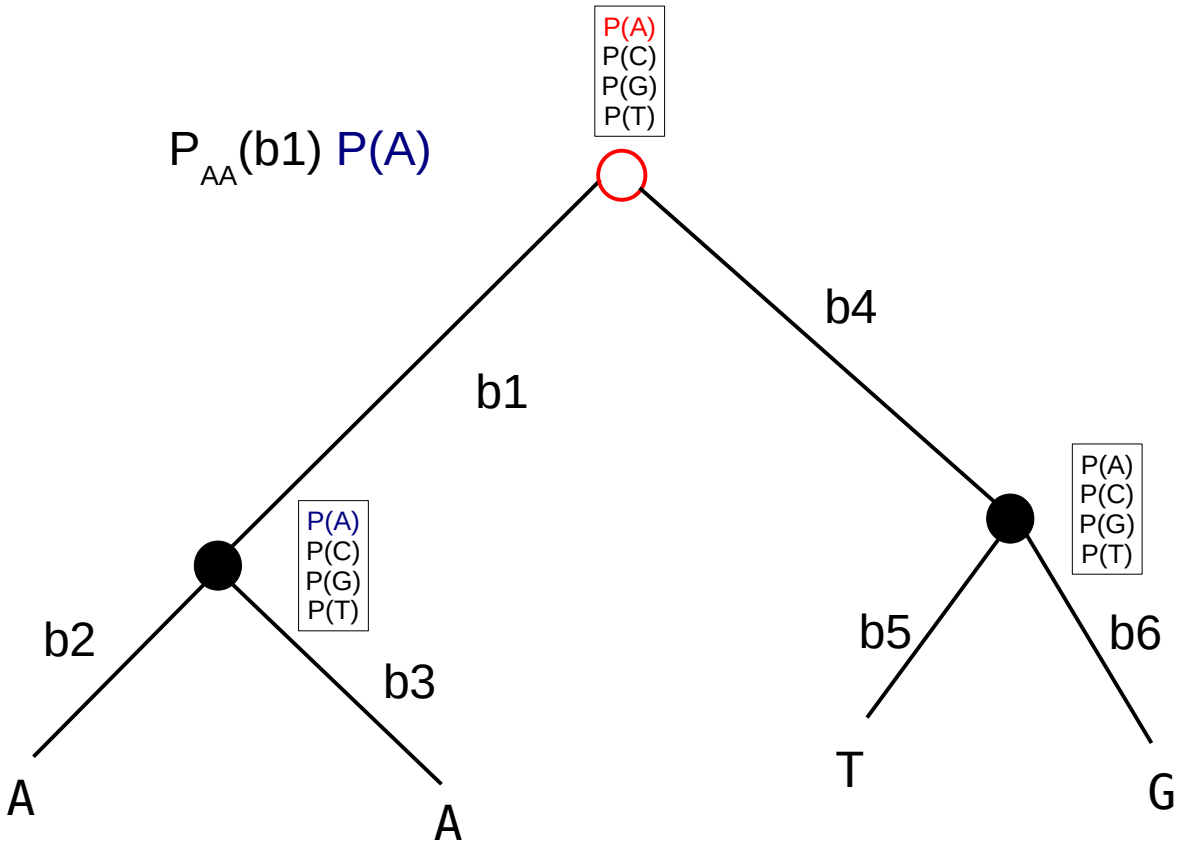


Post order traversal

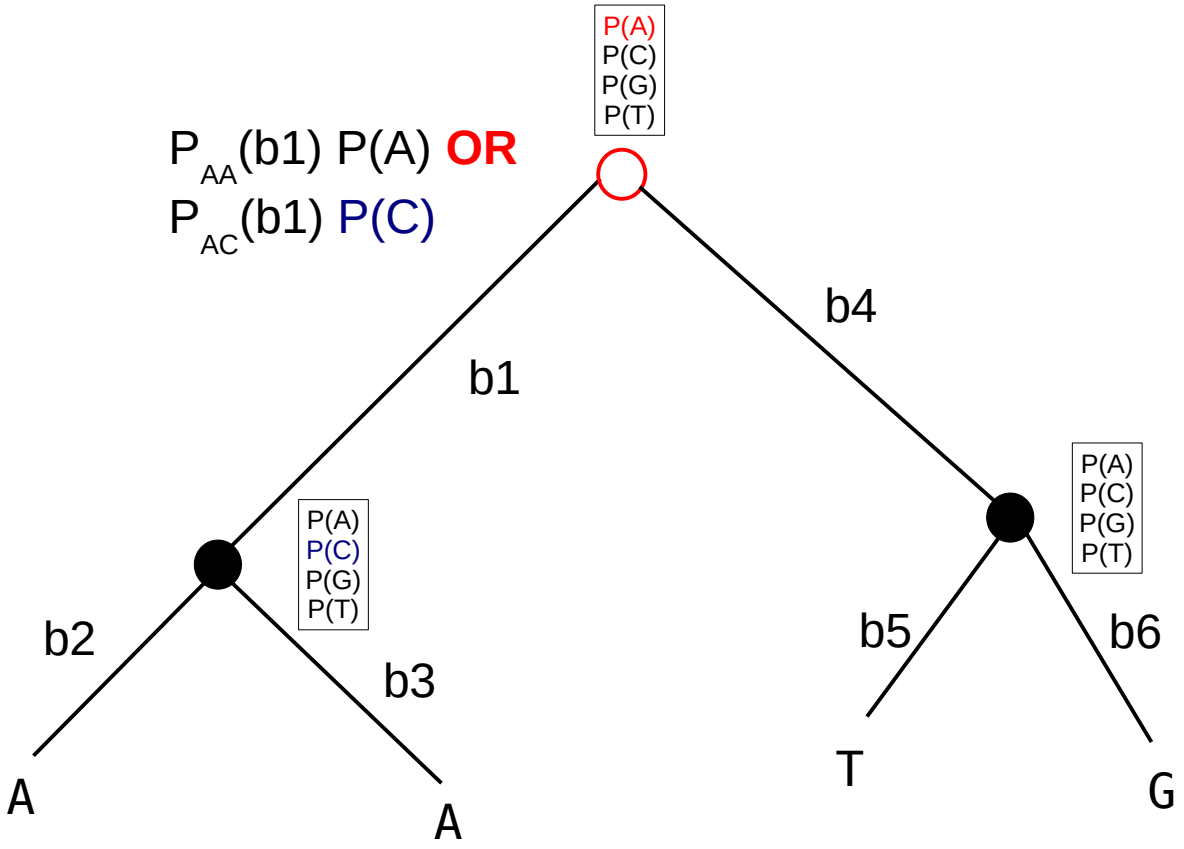
Felsenstein Pruning



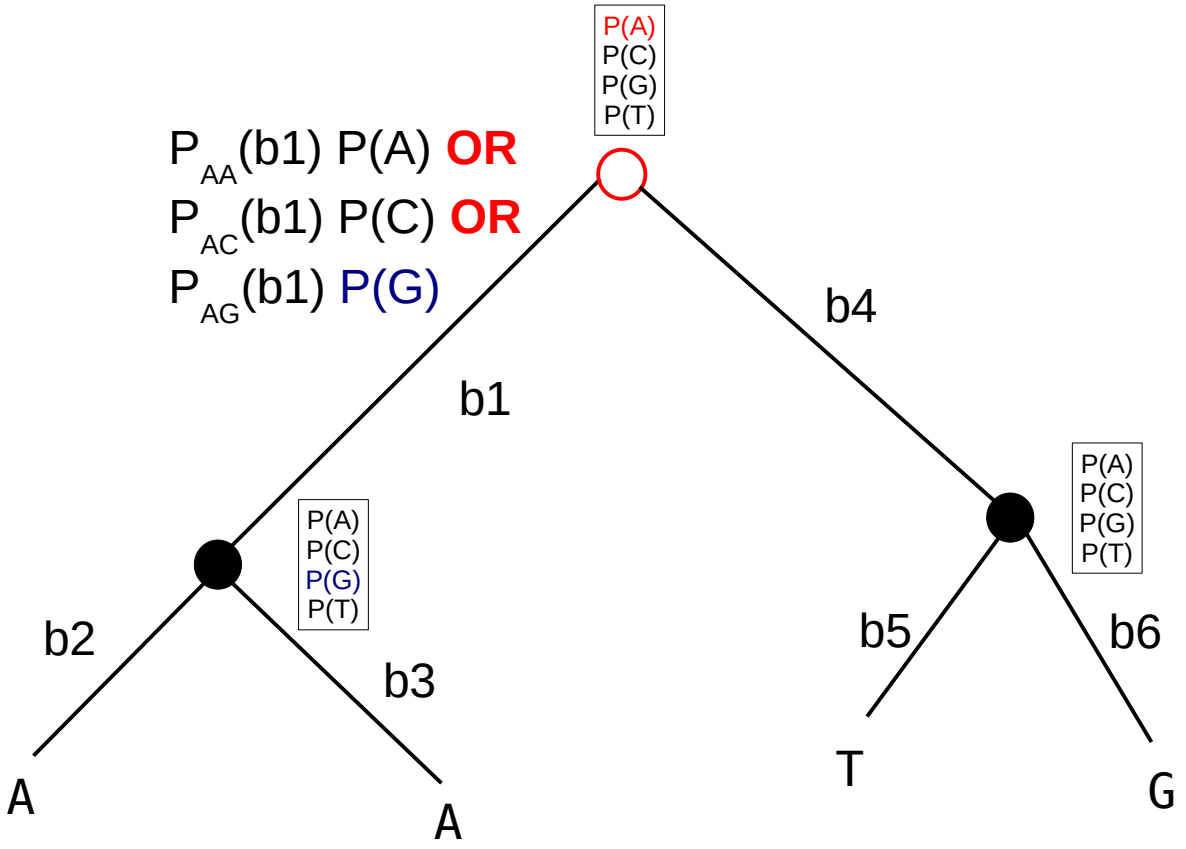
Felsenstein Pruning



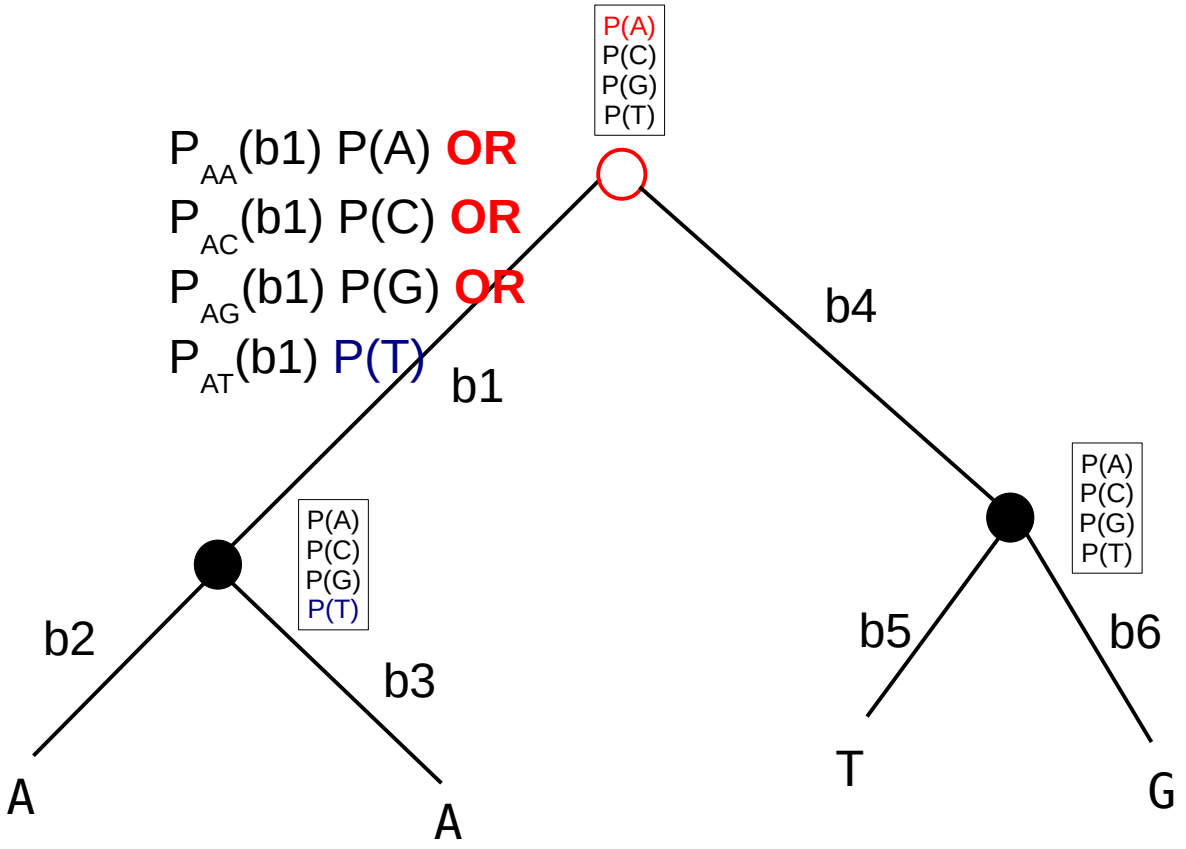
Felsenstein Pruning



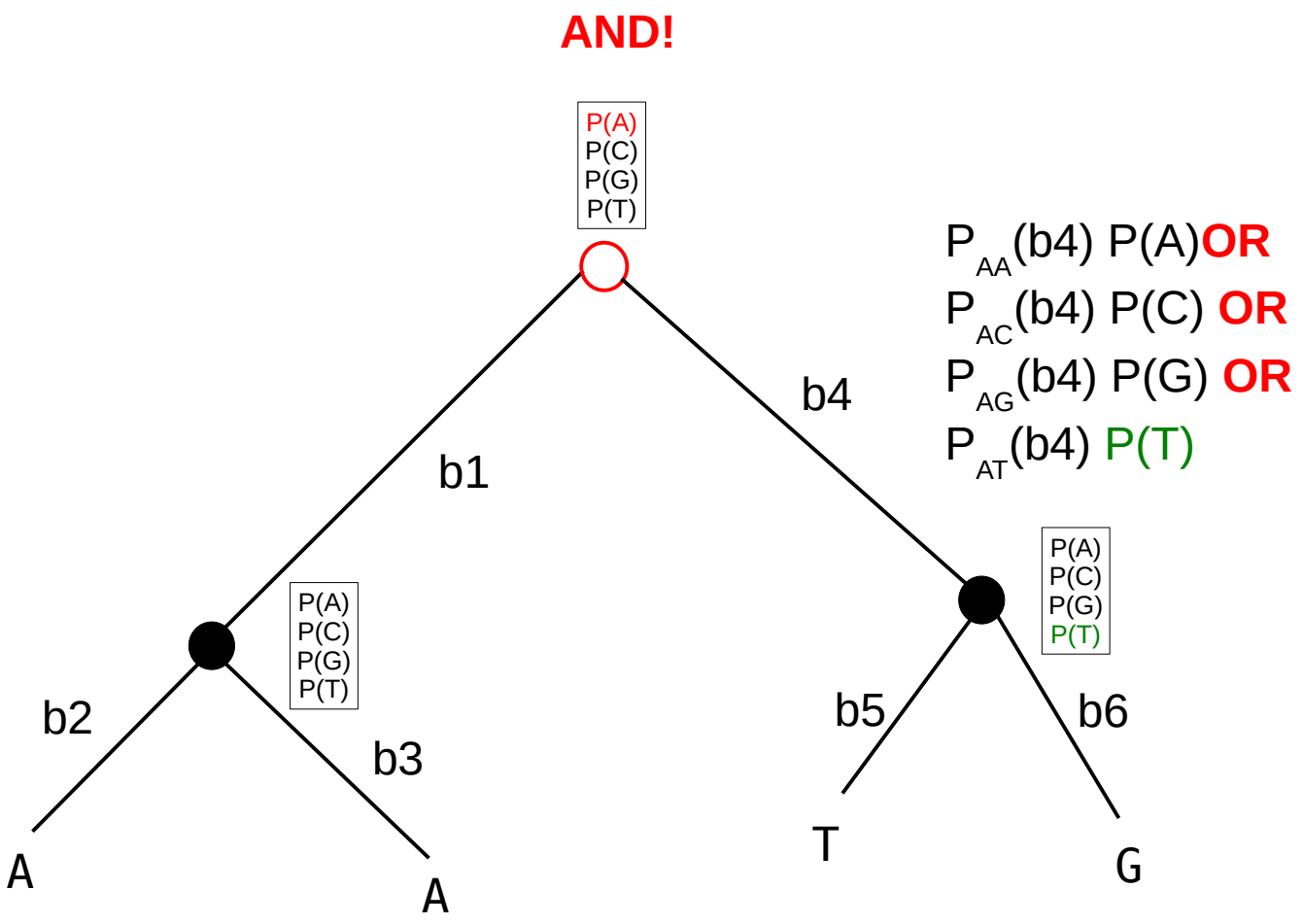
Felsenstein Pruning



Felsenstein Pruning



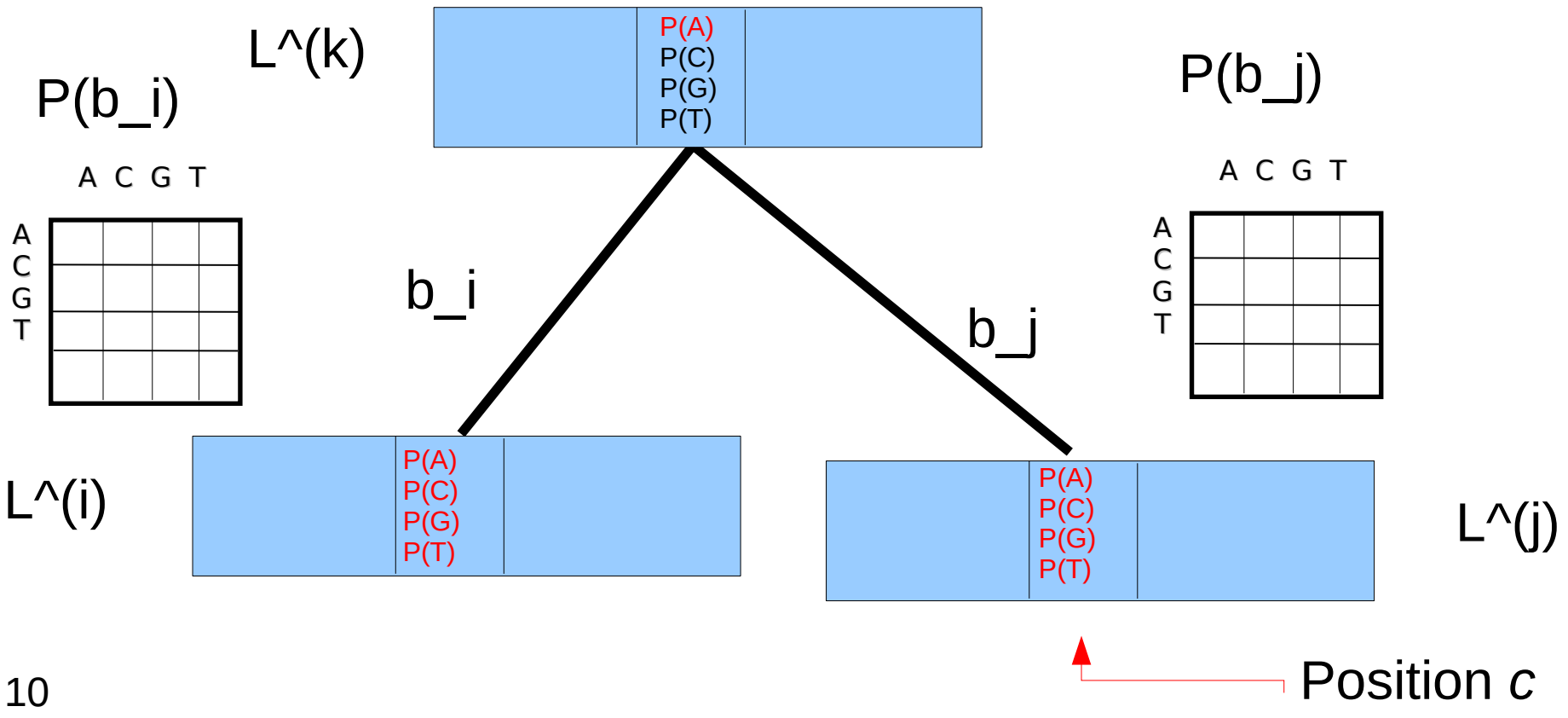
Felsenstein Pruning



Felsenstein Pruning

AND (left branch/right branch)

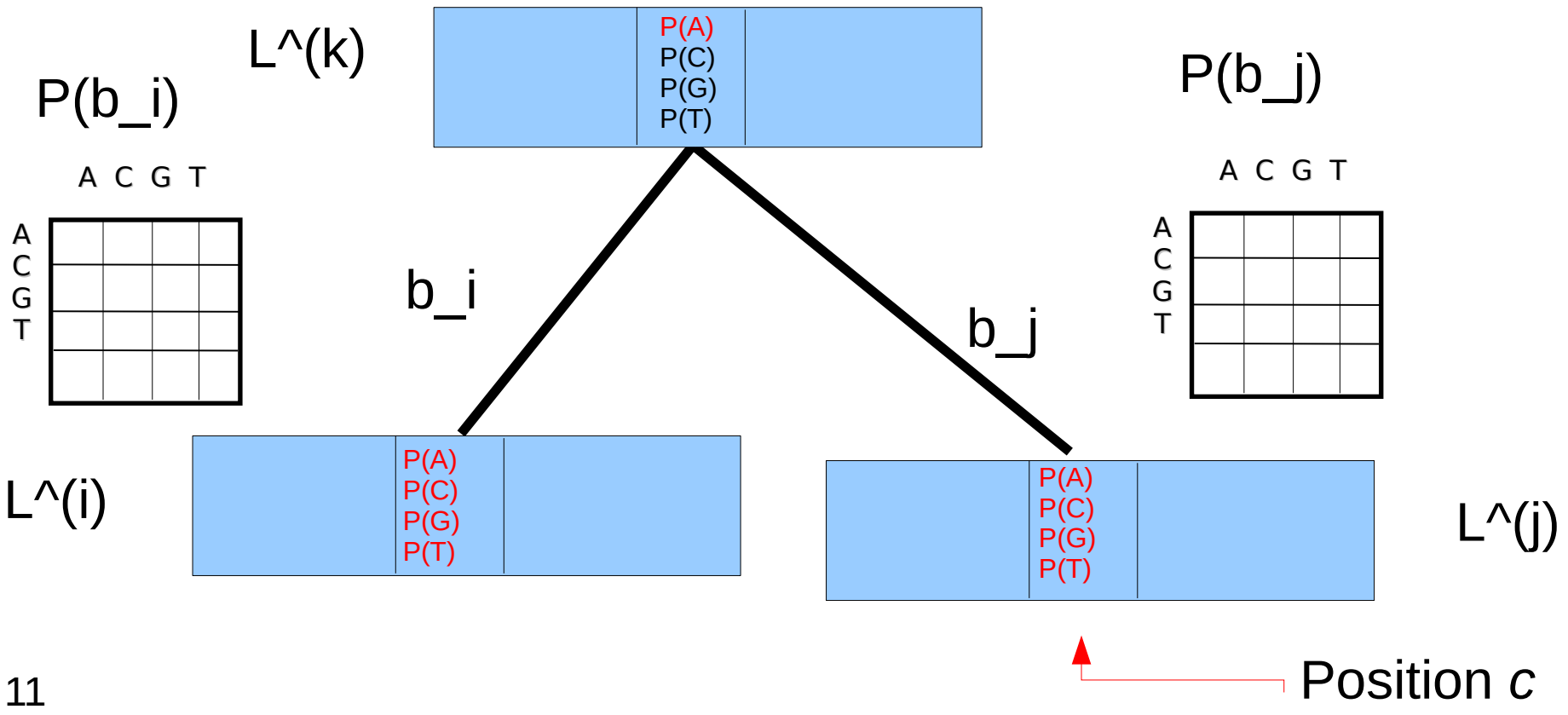
$$\vec{L}_A^{(k)}(c) = \Big(\sum_{S=A}^{T} P_{AS}(b_i)\vec{L}_S^{(i)}(c)\Big)\Big(\sum_{S=A}^{T} P_{AS}(b_j)\vec{L}_S^{(j)}(c)\Big)$$



Felsenstein Pruning

OR (along left branch)

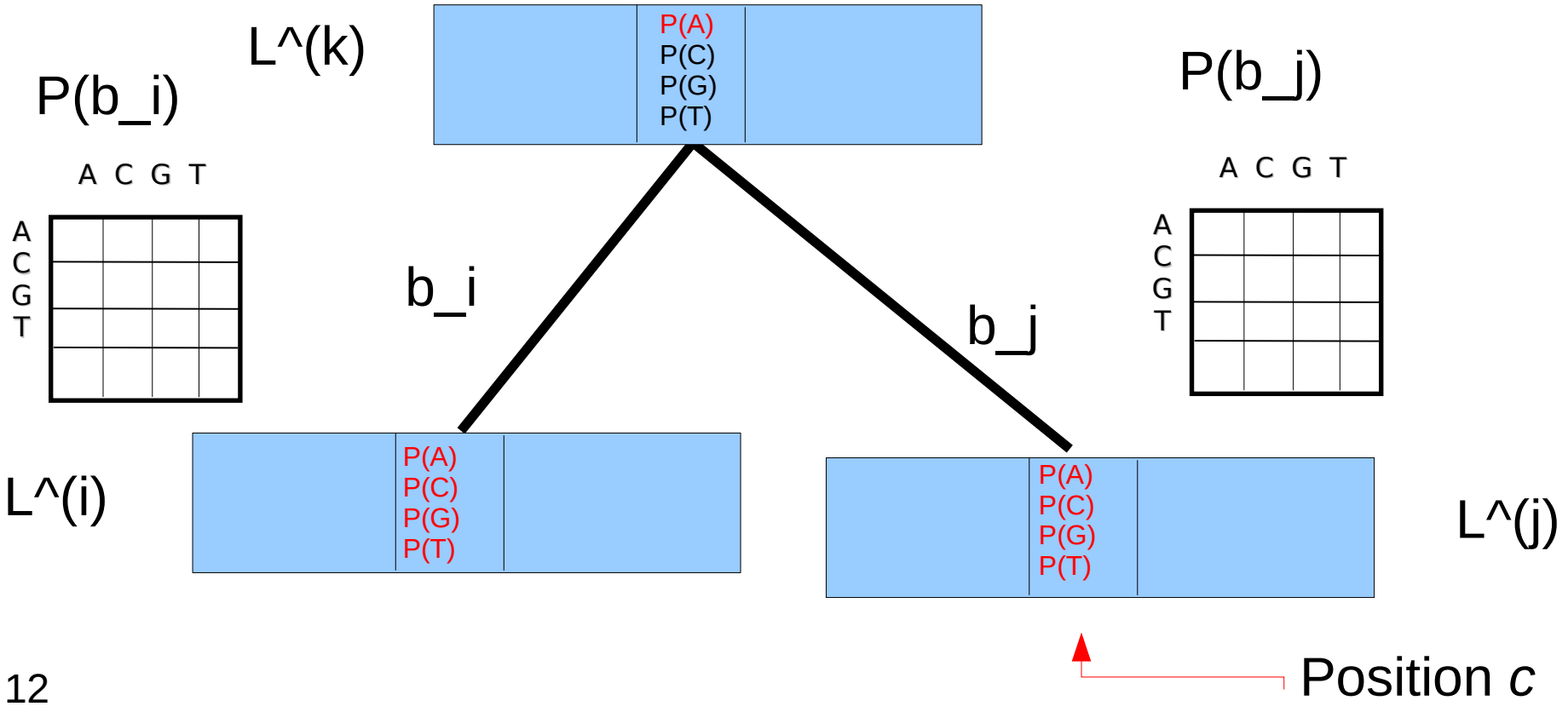
$$\vec{L}_A^{(k)}(c) = \Big(\sum_{S=A}^{T} P_{AS}(b_i)\vec{L}_S^{(i)}(c)\Big)\Big(\sum_{S=A}^{T} P_{AS}(b_j)\vec{L}_S^{(j)}(c)\Big)$$



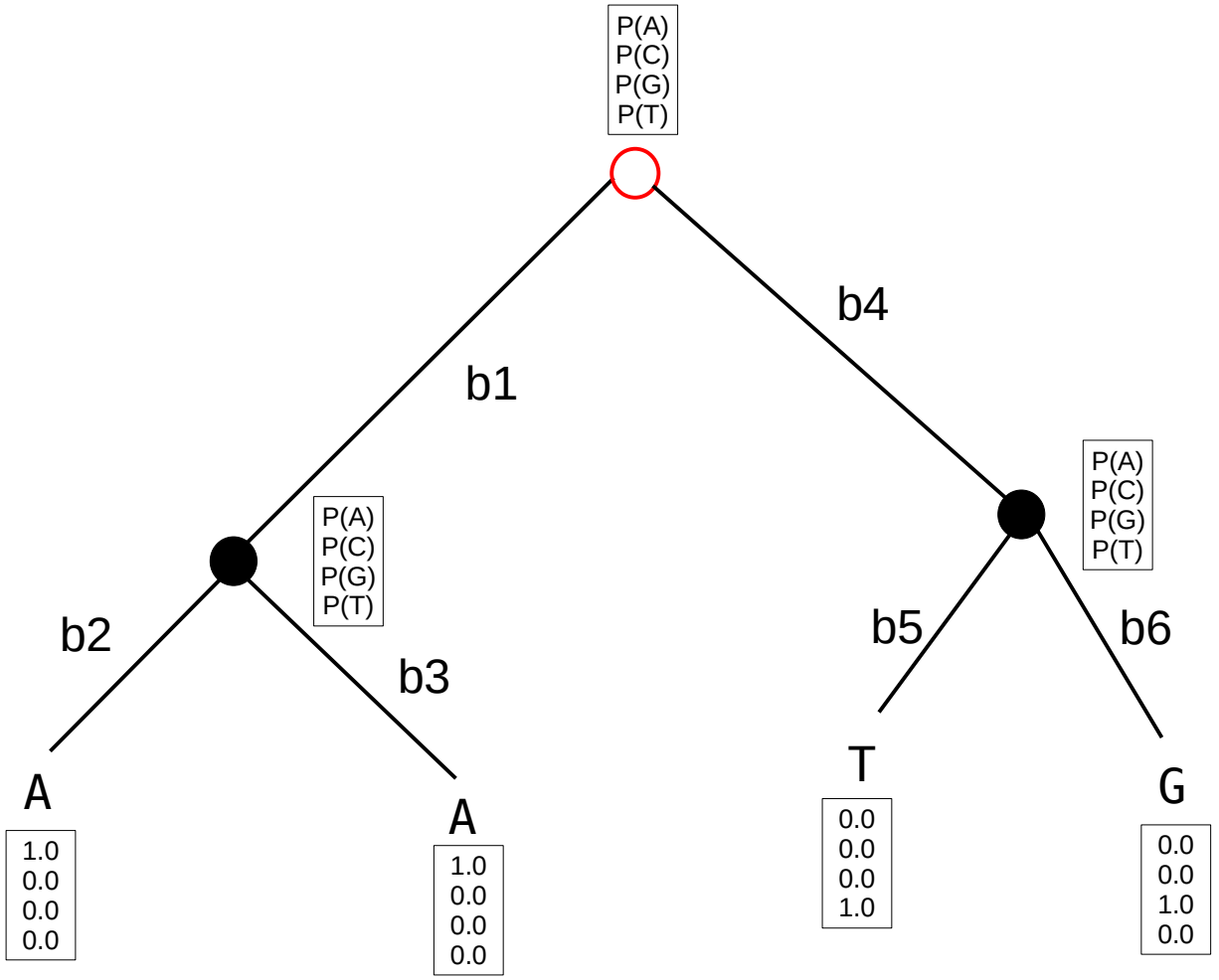
Felsenstein Pruning

OR (along right branch)

$$\vec{L}_A^{(k)}(c) = \Big(\sum_{S=A}^{T} P_{AS}(b_i)\vec{L}_S^{(i)}(c)\Big)\Big(\sum_{S=A}^{T} P_{AS}(b_j)\vec{L}_S^{(j)}(c)\Big)$$

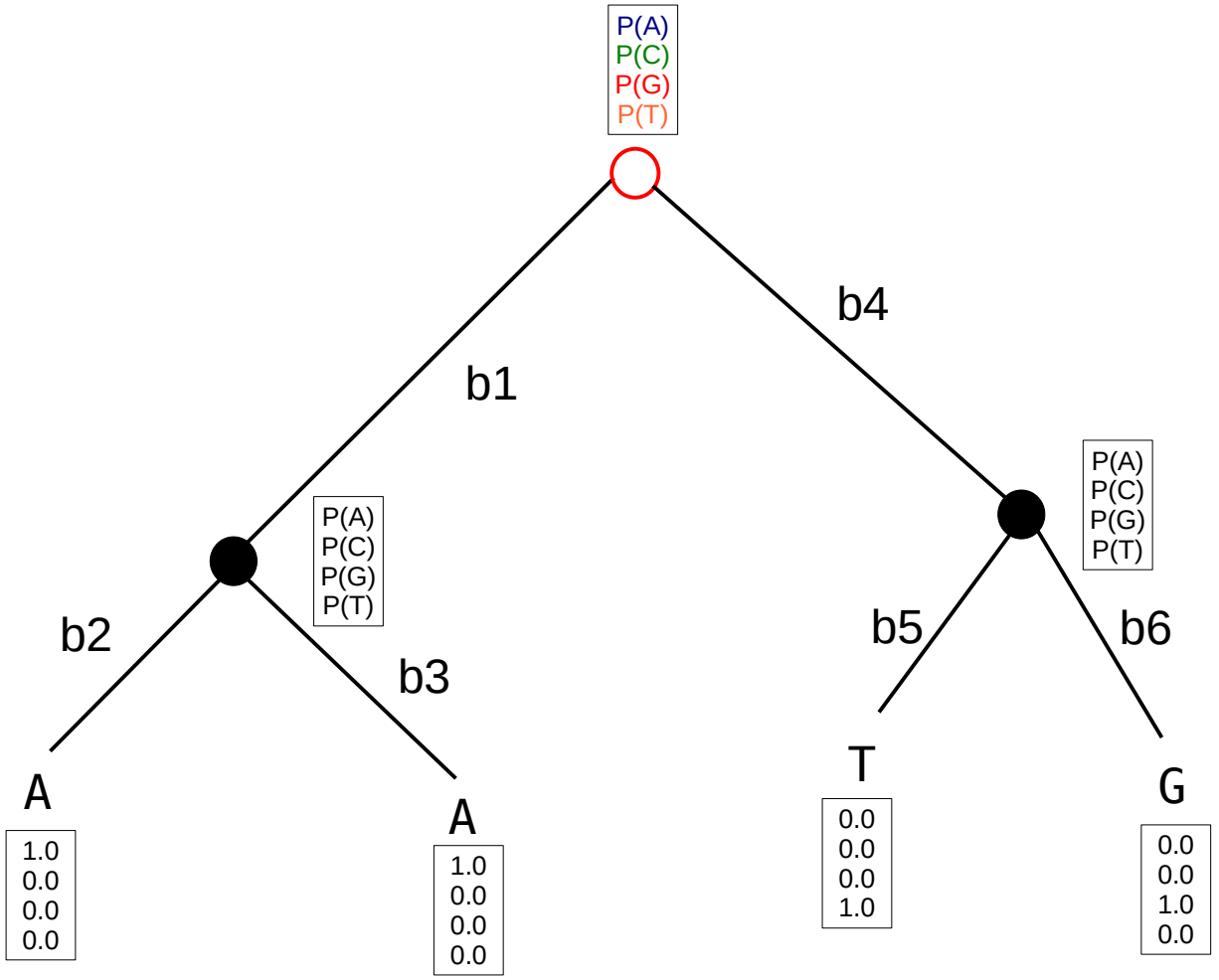


Felsenstein Pruning



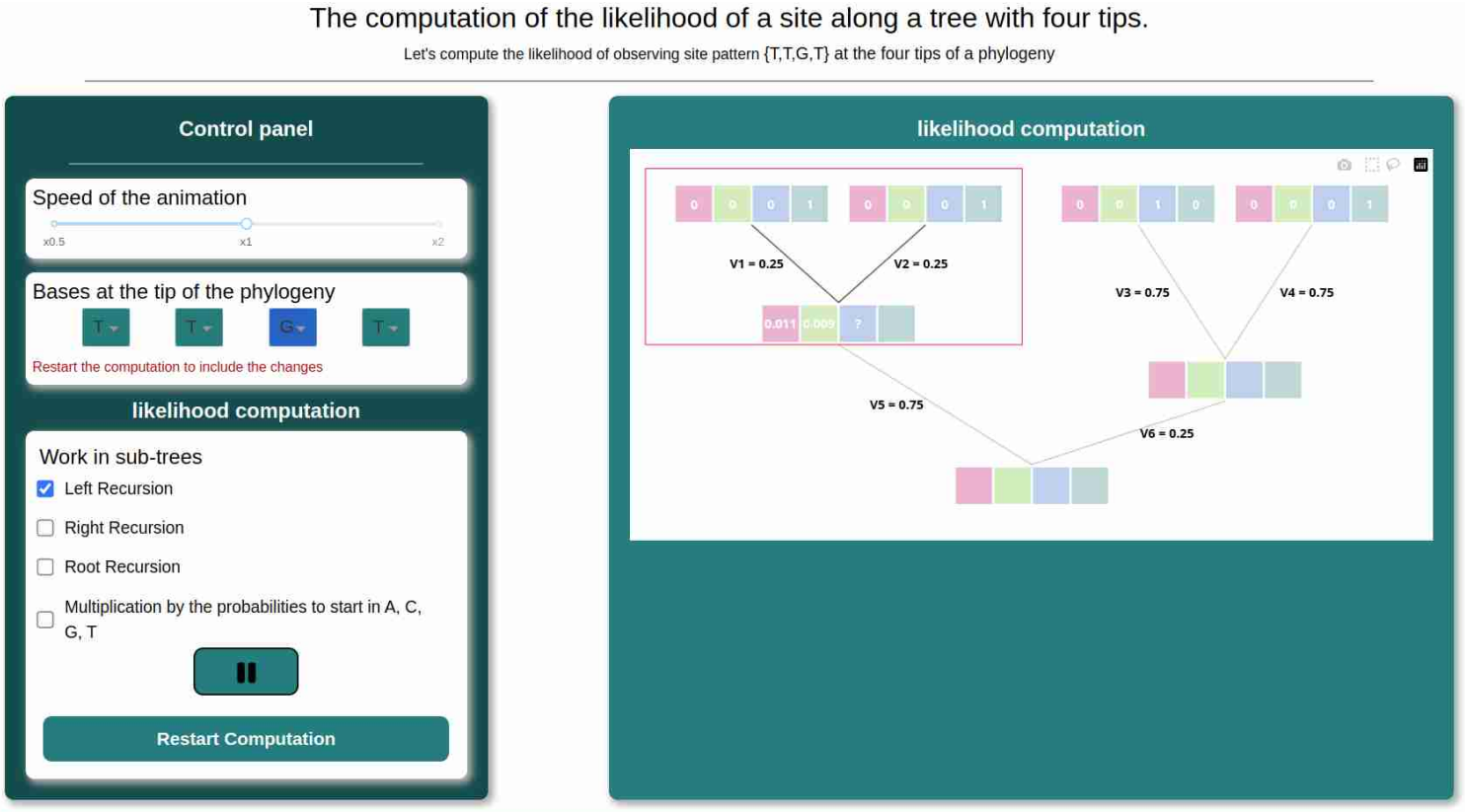
Felsenstein Pruning

Likelihood at the root: $L_i = \pi_A P(A) + \pi_C P(C) + \pi_G P(G) + \pi_T P(T)$



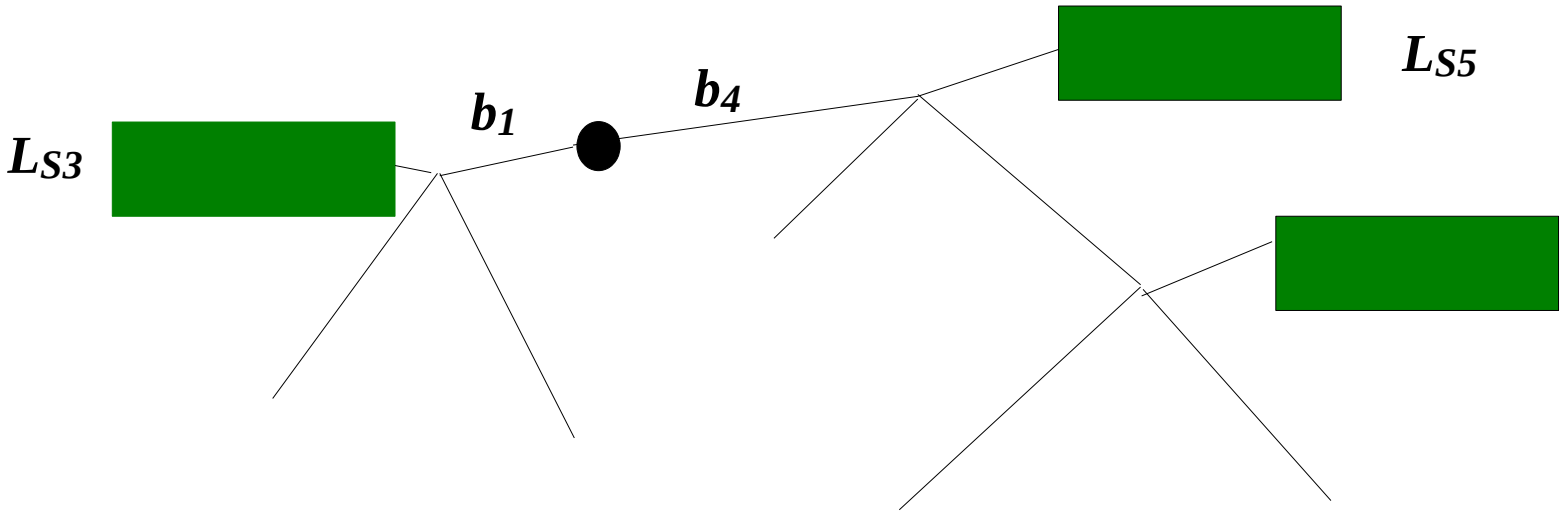
An Excellent Tool to viualize and revise this concept

- https://phylanim.univ-lyon1.fr/LikelihoodTreeComputation



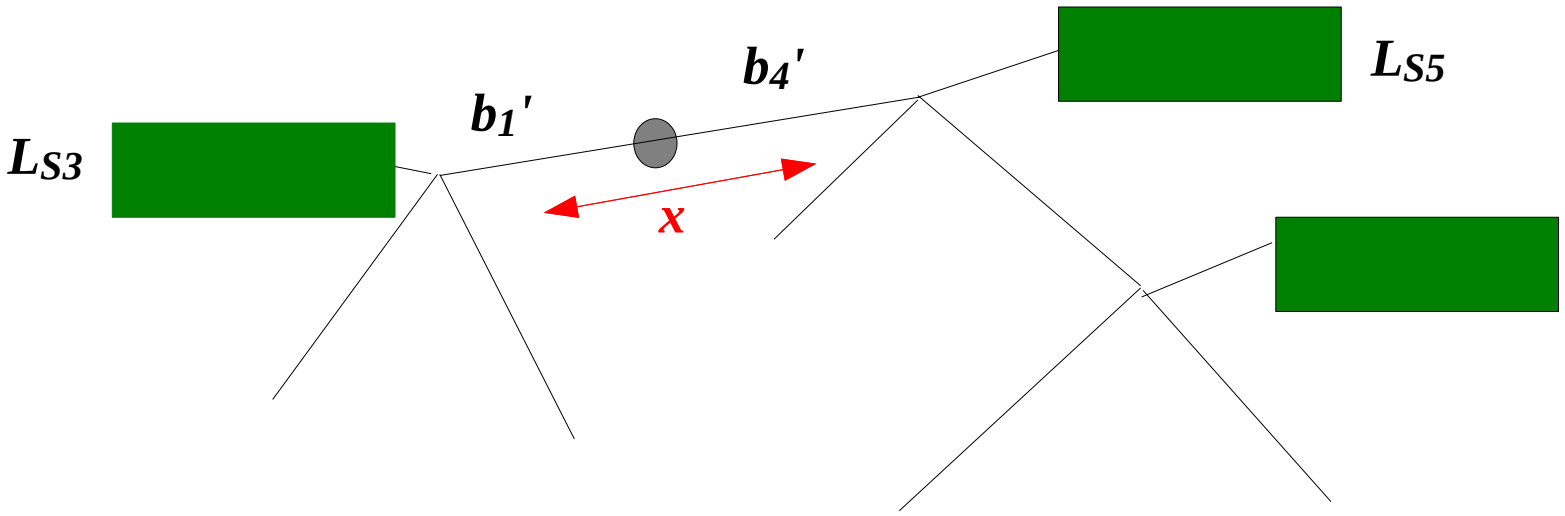
Why is time-reversibility important?

$$L = \sum_{S_4=A}^{T} \pi_{S_4} \sum_{S_3=A}^{T} P_{S_4 S_3}(b_1) L_{S_3}^{(3)} \sum_{S_5=A}^{T} P_{S_4 S_5}(b_4) L_{S_5}^{(5)}$$



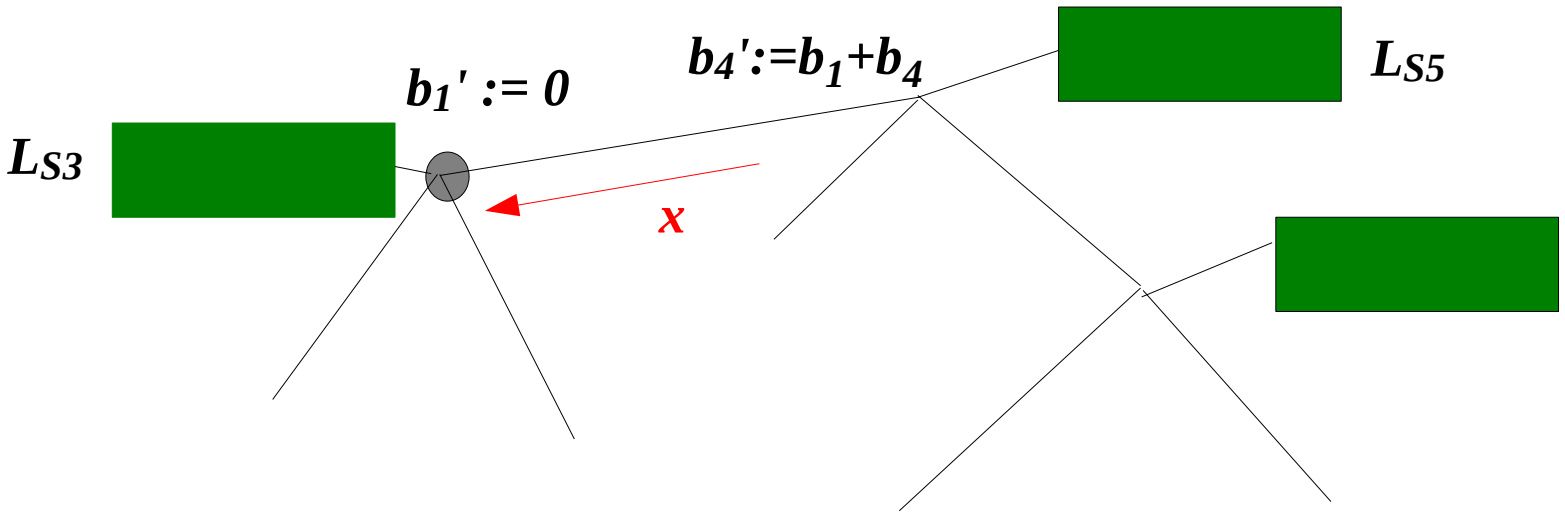
Why is time-reversibility important?

$$L = L' = \sum_{S_4=A}^{T} \pi_{S_4} \sum_{S_3=A}^{T} P_{S_4 S_3}(b_1 + x) L_{S_3}^{(3)} \sum_{S_5=A}^{T} P_{S_4 S_5}(b_4 - x) L_{S_5}^{(5)}$$



Why is time-reversibility important?

$$L = L' = \sum_{S_4=A}^{T} \pi_{S_4} \sum_{S_3=A}^{T} P_{S_4 S_3}(b_1 + x) L_{S_3}^{(3)} \sum_{S_5=A}^{T} P_{S_4 S_5}(b_4 - x) L_{S_5}^{(5)}$$



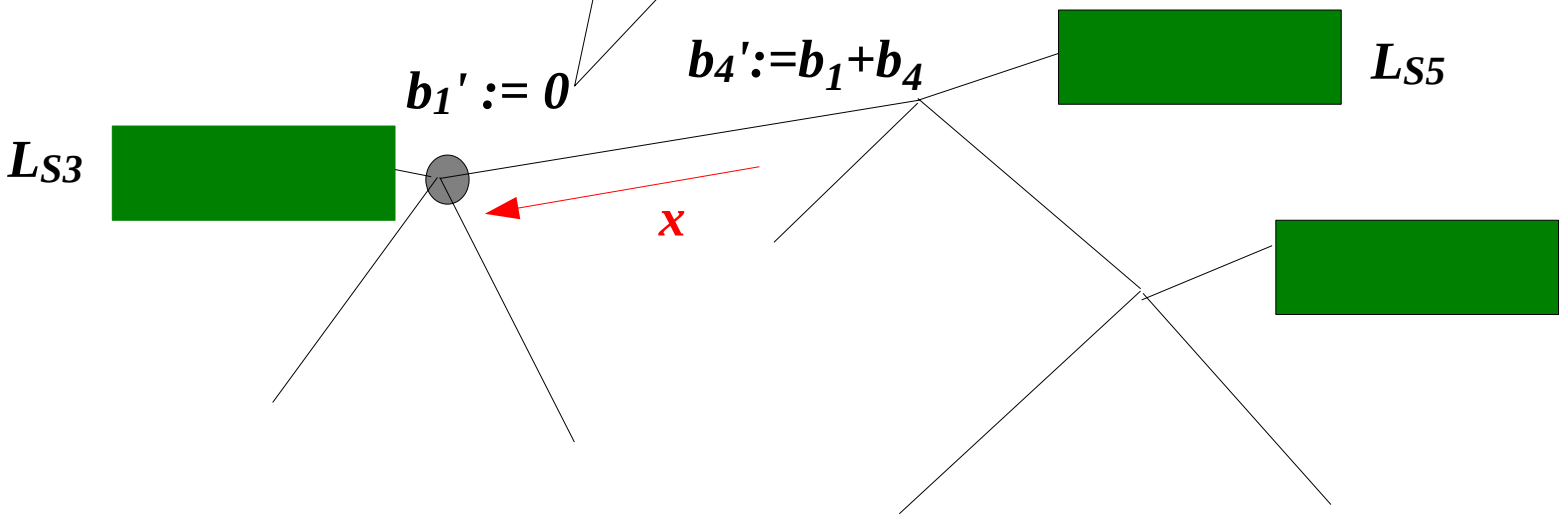
Why is time-reversibility important?

This observation can be applied recursively to the tree

→

It does not matter at all where we place the root!

$$L = L' = \sum_{S_4=A}^{T} \qquad \sum_{S_5=A}^{T} P_{S_4 S_5}(b_4 - x) L_{S_5}^{(5)}$$

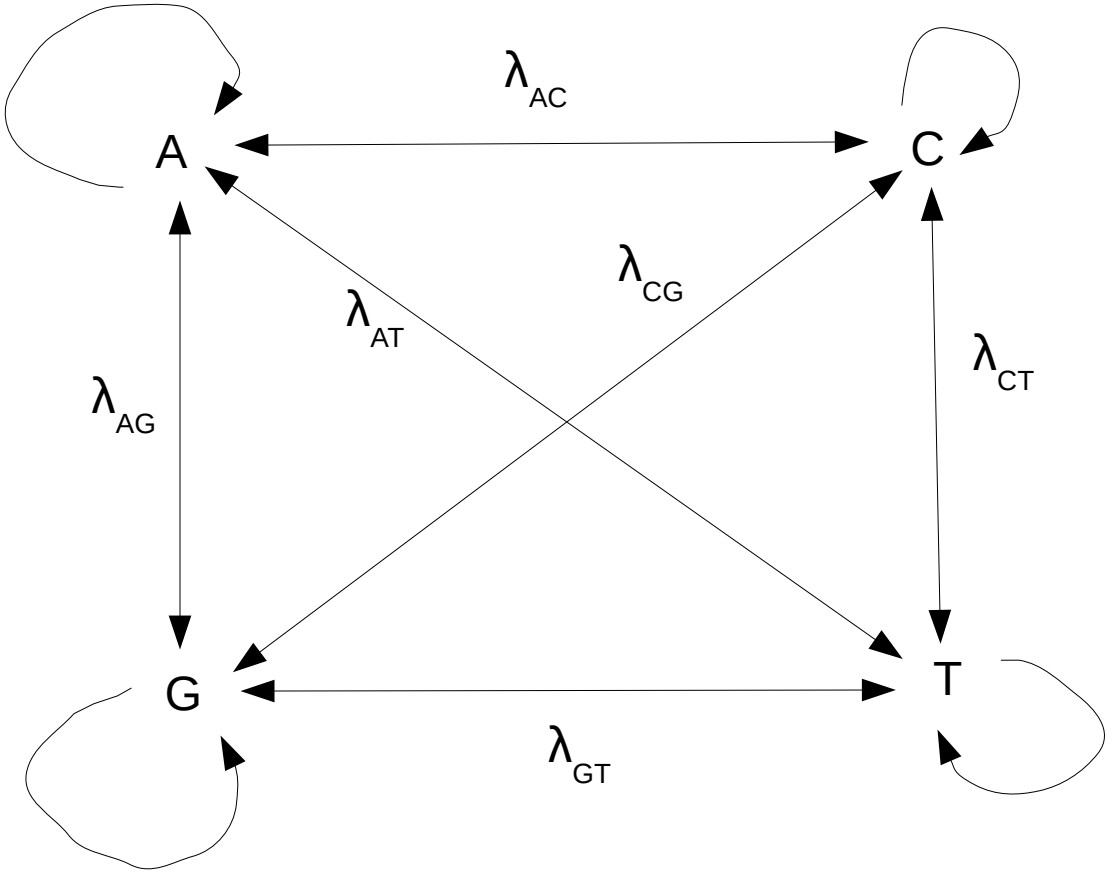


Outline

- Last time:
  - How to Compute the Likelihood of a tree
  - How to compute the Likelihood efficiently: Felsenstein Pruning Algorithm
- Today & next time
  - **What is hidden in *P(t)* – what do the models look like?**
  - How to compute the Maximum Likelihood score on a tree?
  - Advanced substitution models
  - Efficiently computing the Likelihood on trees
  - Parallel Likelihood computations

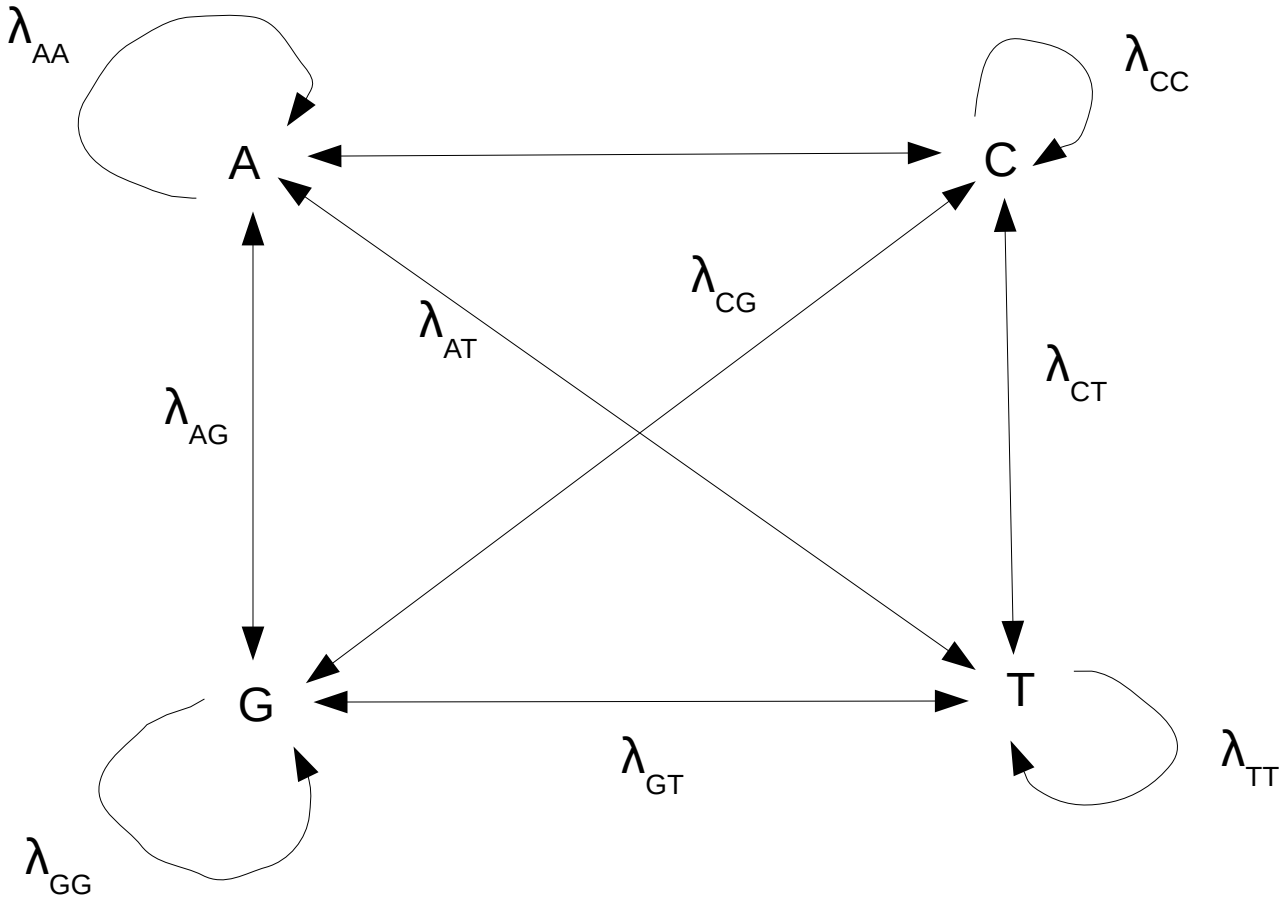
What's in the black box $P_{ij}(t)$?

Instantaneous rate matrix *R*!



What's in the black box $P_{ij}(t)$?

What about the probabilities of staying in the current state?
→ they are given by the properties of continuous Markov chains!
e.g., $\lambda_{AA} = -(\lambda_{AC} + \lambda_{AG} + \lambda_{AT})$ → remember from lecture on Markov models:
rows in the *R* matrix need to sum to **0**



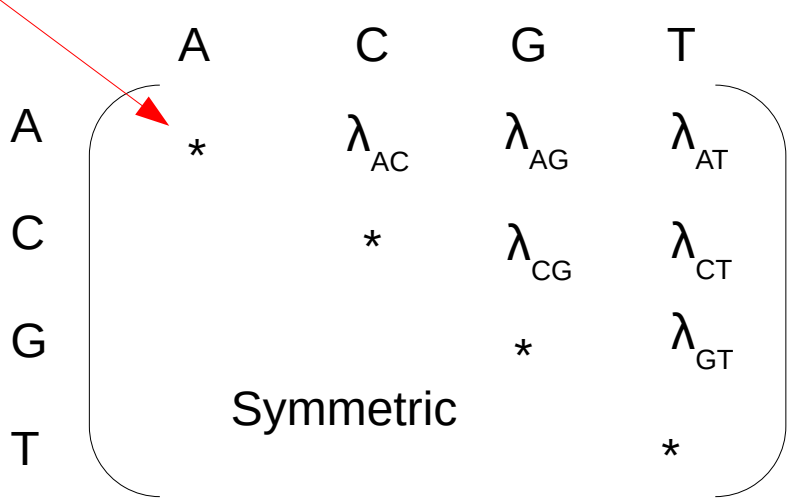
What's in the black box $P_{ij}(t)$?

	A	C	G	T
A	*	λ_{AC}	λ_{AG}	λ_{AT}
C		*	λ_{CG}	λ_{CT}
G			*	λ_{GT}
T		Symmetric		*

What's in the black box $P_{ij}(t)$?

Diagonal values are given by the off-diagonal values (R matrix property)

$\lambda_{AA} = -(\lambda_{AC} + \lambda_{AG} + \lambda_{AT})$

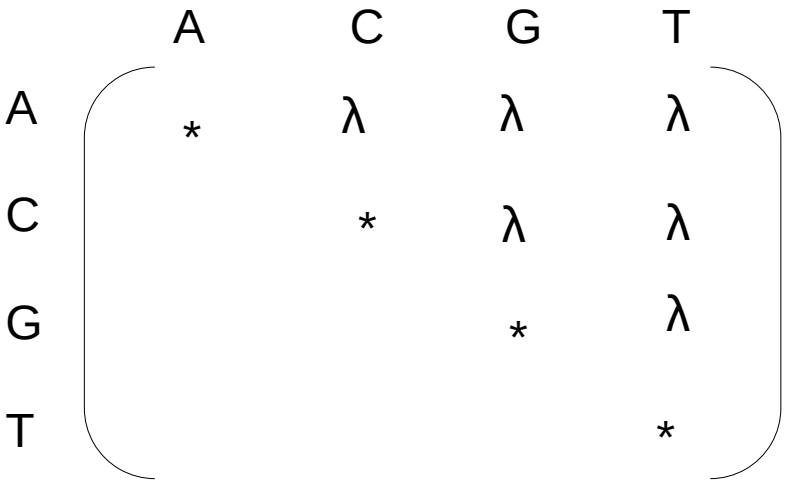


What's in the black box $P_{ij}(t)$?

	A	C	G	T
A	*	λ_{AC}	λ_{AG}	λ_{AT}
C		*	λ_{CG}	λ_{CT}
G			*	λ_{GT}
T		Symmetric		*

Equilibrium frequency vector $\pi = (\pi_A, \pi_C, \pi_G, \pi_T)$ *where* $\pi_A + \pi_C + \pi_G + \pi_T = 1$

The simple Jukes-Cantor model



П = (1/4, 1/4, 1/4, 1/4)

The Felsenstein 81 model

	A	C	G	T
A	*	λ	λ	λ
C		*	λ	λ
G			*	λ
T				*

$\Pi_i \neq \Pi_j$

Kimura 2-parameter model 1980

	A	C	G	T
A	*	λ	ζ	λ
C		*	ζ	λ
G			*	ζ
T				*

П = (1/4, 1/4, 1/4, 1/4)

HKY85

	A	C	G	T
A	*	λ	ζ	λ
C		*	ζ	λ
G			*	ζ
T				*

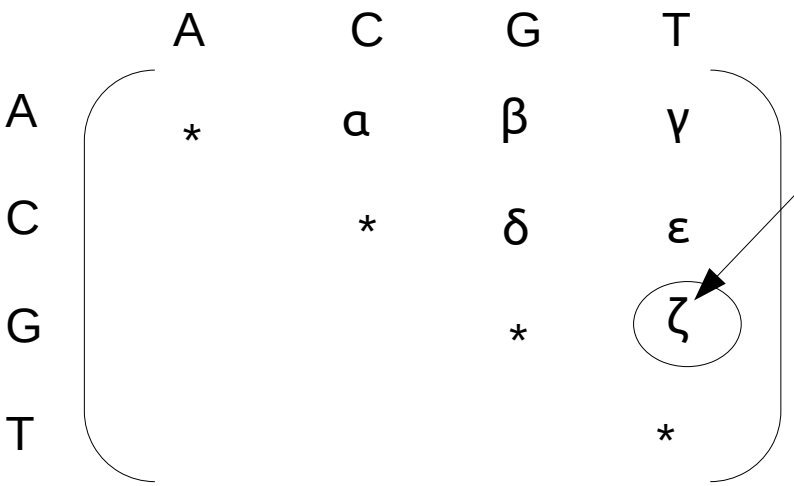
$\Pi_i \neq \Pi_j$

GTR 1986

	A	C	G	T
A	*	α	β	γ
C		*	δ	ε
G			*	ζ
T				*

$\Pi_i \neq \Pi_j$

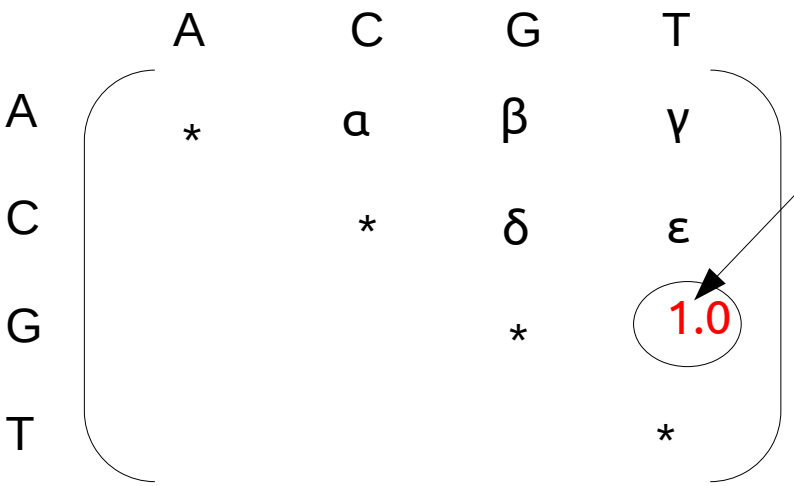
GTR 1986



Note that these are **relative** rates, their values only matter relative to each other, so we can set $\zeta := 1.0$ by default

$\Pi_i \neq \Pi_j$

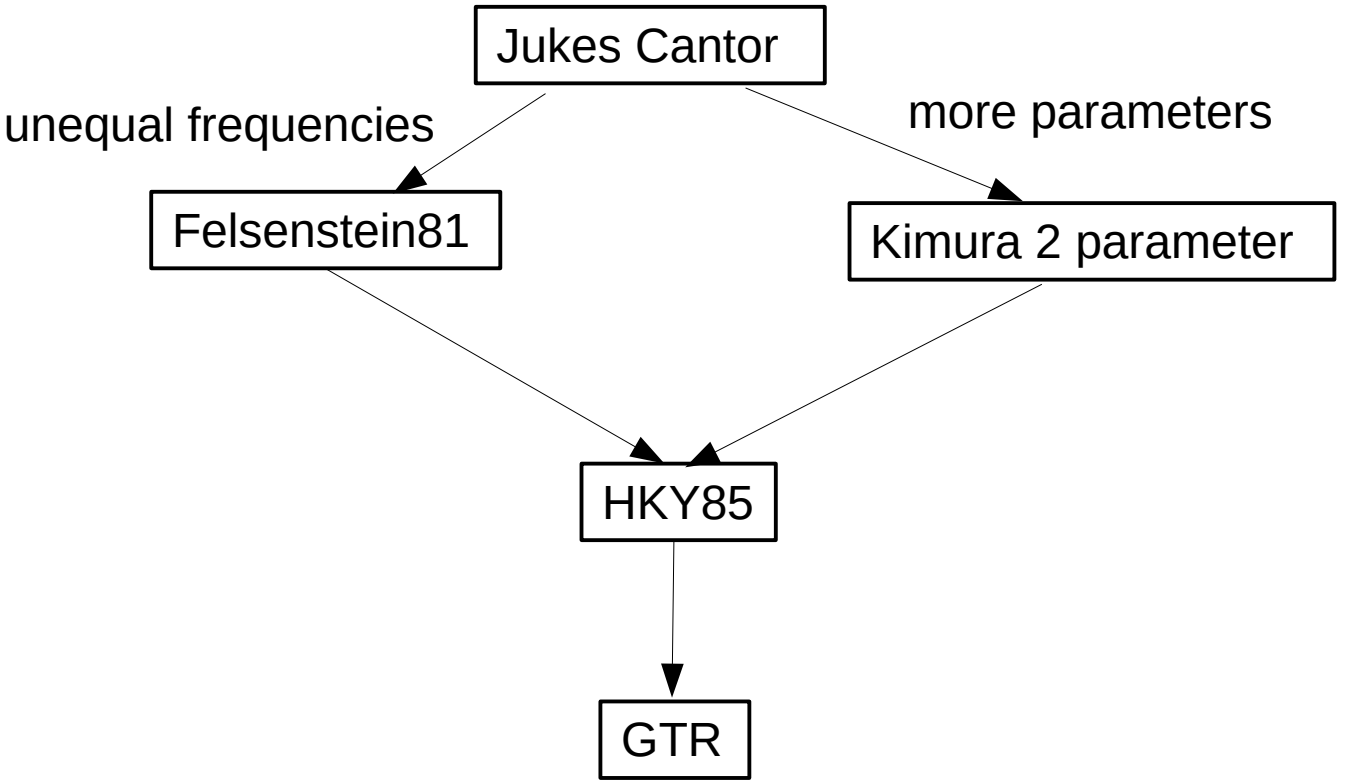
GTR 1986



$\Pi_i \neq \Pi_j$

Note that these are **relative** rates, their values only matter relative to each other, so we can set $\zeta := 1.0$ by default. Although the GTR model has 6 rates, it only has 5 free parameters!

Model Hierarchy



GTR 1986

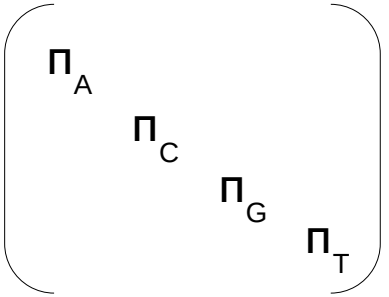
This is a rate matrix, time reversibility would require $\pi_i r_{ij} = \pi_j r_{ji}$

$\Pi_i \neq \Pi_j$

GTR 1986

This is a rate matrix, time reversibility would require $\pi_i r_{ij} = \pi_j r_{ji}$

Solution: introduce a Q matrix $Q := diag(\pi)\ R$



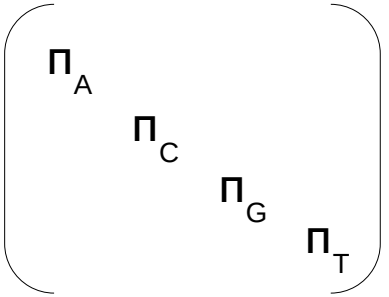
$\Pi_i \neq \Pi_j$

GTR 1986

$\Pi_i \neq \Pi_j$

This is a rate matrix, time reversibility would require $\pi_i r_{ij} = \pi_j r_{ji}$

Solution: introduce a Q matrix $Q := diag(\pi)\ R$



Then, $\pi_i r_{ij} = \pi_j r_{ji}$ holds

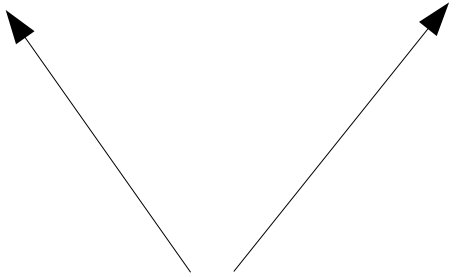
So how do we compute P(t) from Q?

- As we have seen in the lecture on Markov chains:

  $P(t) = e^{Qt} = I + Qt + 1/2!\,(Qt)^2 + 1/3!\,(Qt)^3 + \ldots$

- but this is unfortunately a matrix exponential :-(
- I will spare you the details, but in general, e.g., for GTR we need to apply an Eigenvector/Eigenvalue decomposition of *Q* to calculate:

  $P(t) = U\, exp(diag(\lambda_i)t)\, U^{-1}$



Matrix and inverse matrix of eigenvectors of *Q*

So how do we compute P(t) from Q?

- As we have seen in the lecture on Markov chains:

  $P(t) = e^{Qt} = I + Qt + 1/2! (Qt)^2 + 1/3! (Qt)^3 + \ldots$

- but this is unfortunately a matrix exponential :-(
- I will spare you the details, but in general, e.g., for GTR we need to apply an Eigenvector/Eigenvalue decomposition of *Q* to calculate:

  $P(t) = U \exp(\mathrm{diag}(\lambda_i)t)\, U^{-1}$

Diagonal matrix of eigenvalues of *Q*, here the exponential function *exp()* is invoked on scalar values!

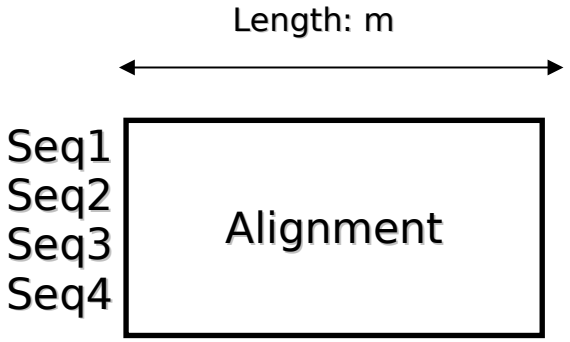
Likelihood Calculations

- So far, we have only seen how to calculate **a** likelihood on a
  - given, fixed tree topology
  - with given fixed branch lengths
  - and given, fixed remaining model parameters
- Computing the **maximum** likelihood score, is much more complicated as it requires
  1. functions for optimizing continuous parameters
  2. functions for searching the discrete space of trees

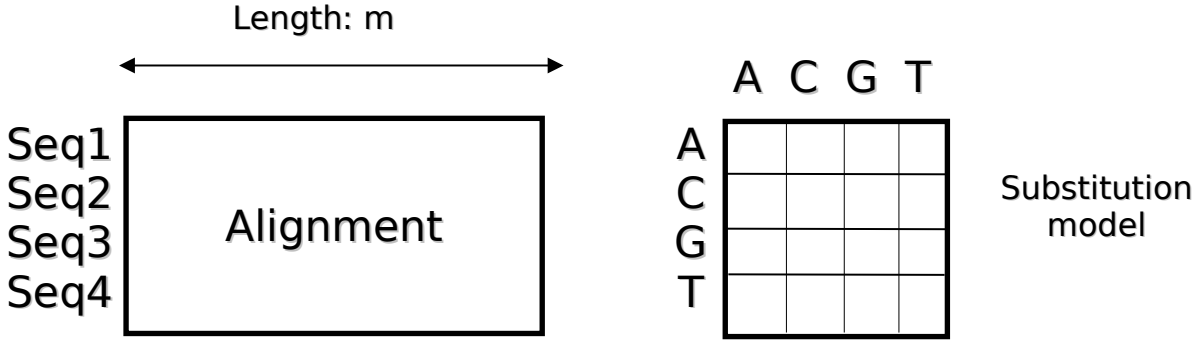
Outline

- Last time:
  - How to Compute the Likelihood of a tree
  - How to compute the Likelihood efficiently: Felsenstein Pruning Algorithm
- Today & next time
  - What is hidden in P(t) – what do the models look like ?
  - **How to compute the Maximum Likelihood score on a tree?**
  - Advanced substitution models
  - Efficiently computing the Likelihood on trees
  - Parallel Likelihood computations

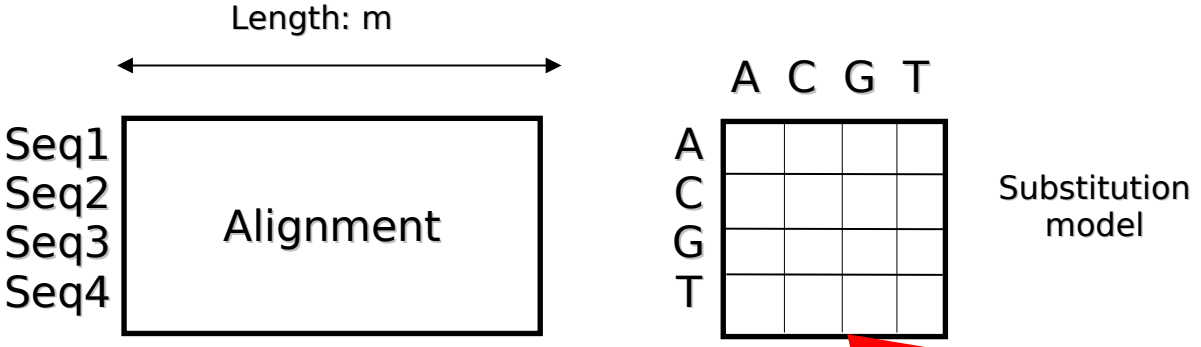
Maximum Likelihood



Maximum Likelihood

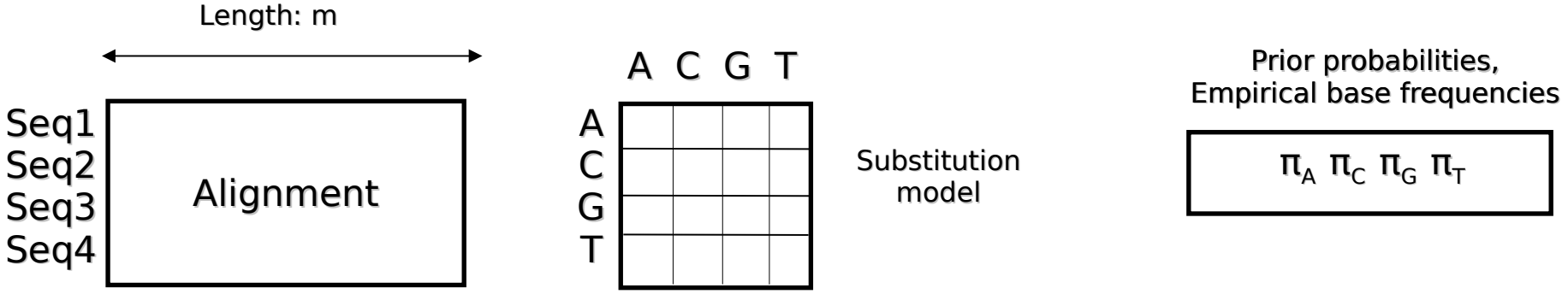


Maximum Likelihood

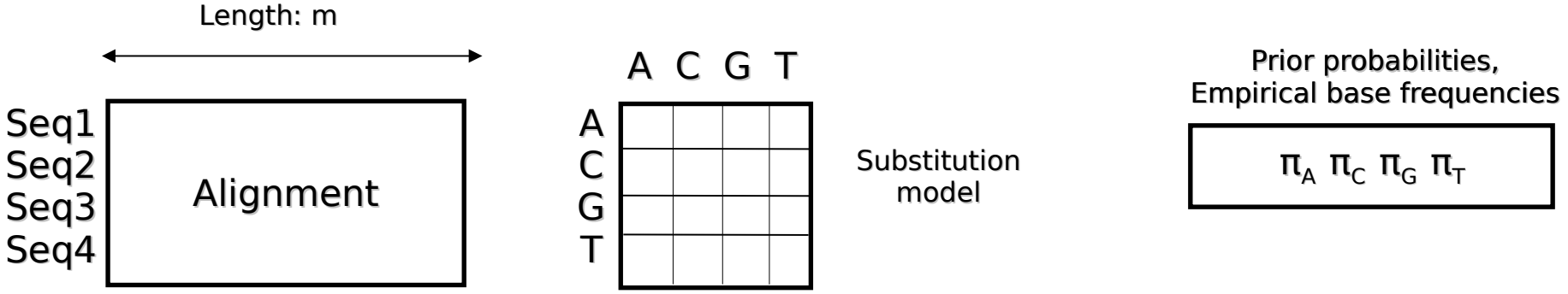


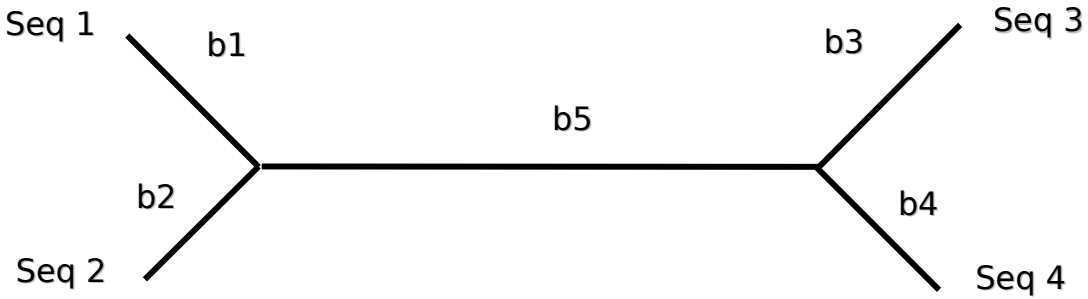
Commonly denoted as *Q* matrix: transition probs for time *dt*, for time *t*: $P(t)=e^{Qt}$

Maximum Likelihood

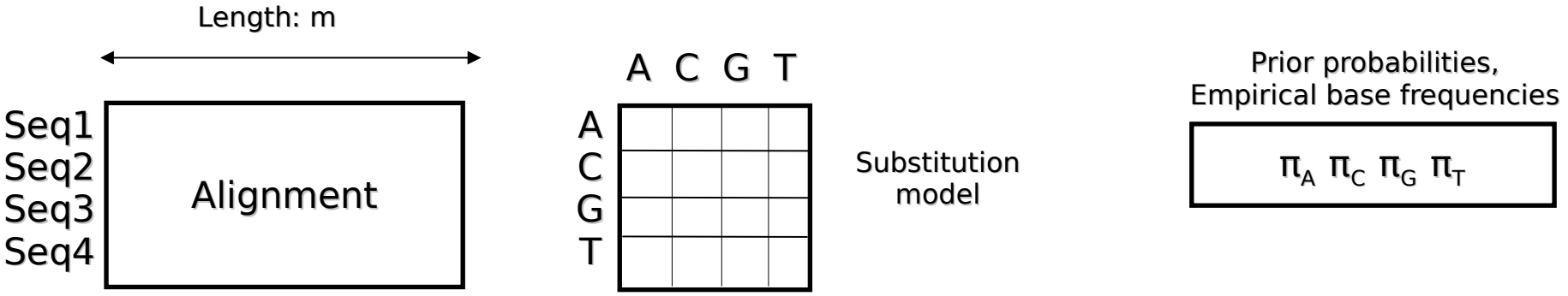


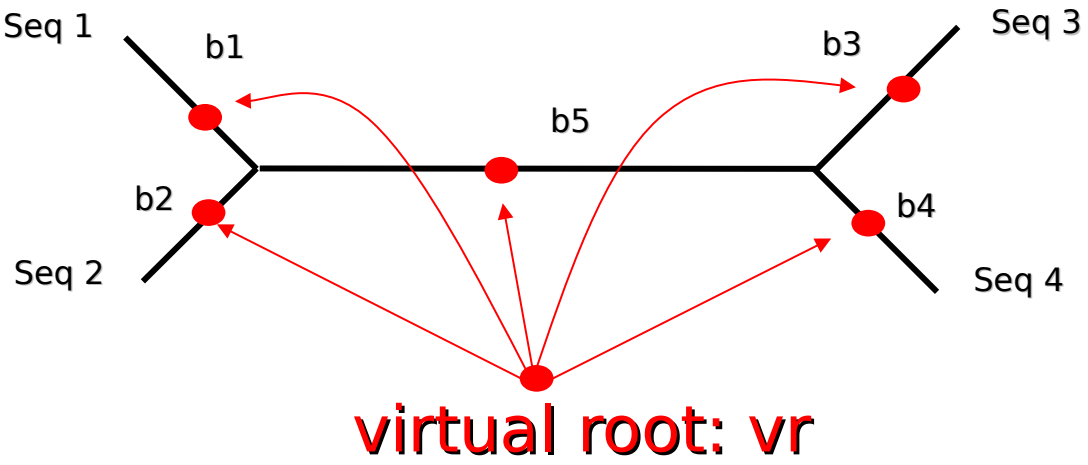
Maximum Likelihood



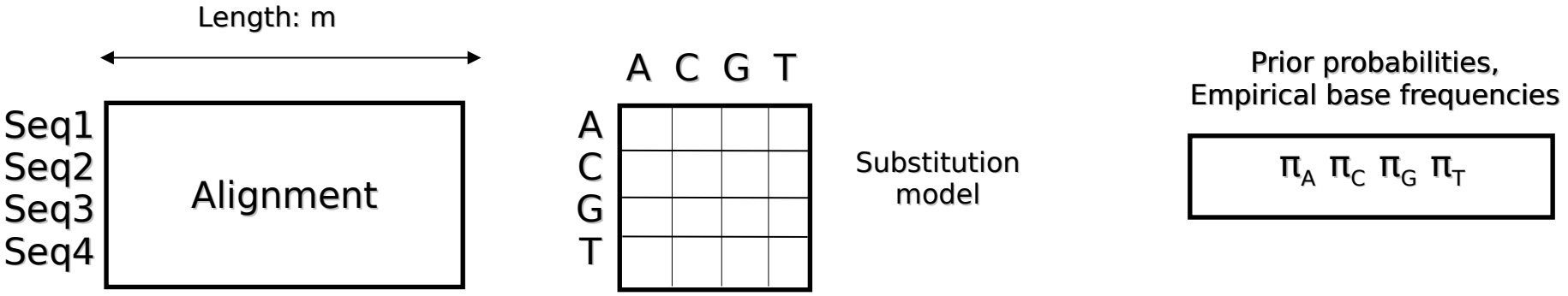


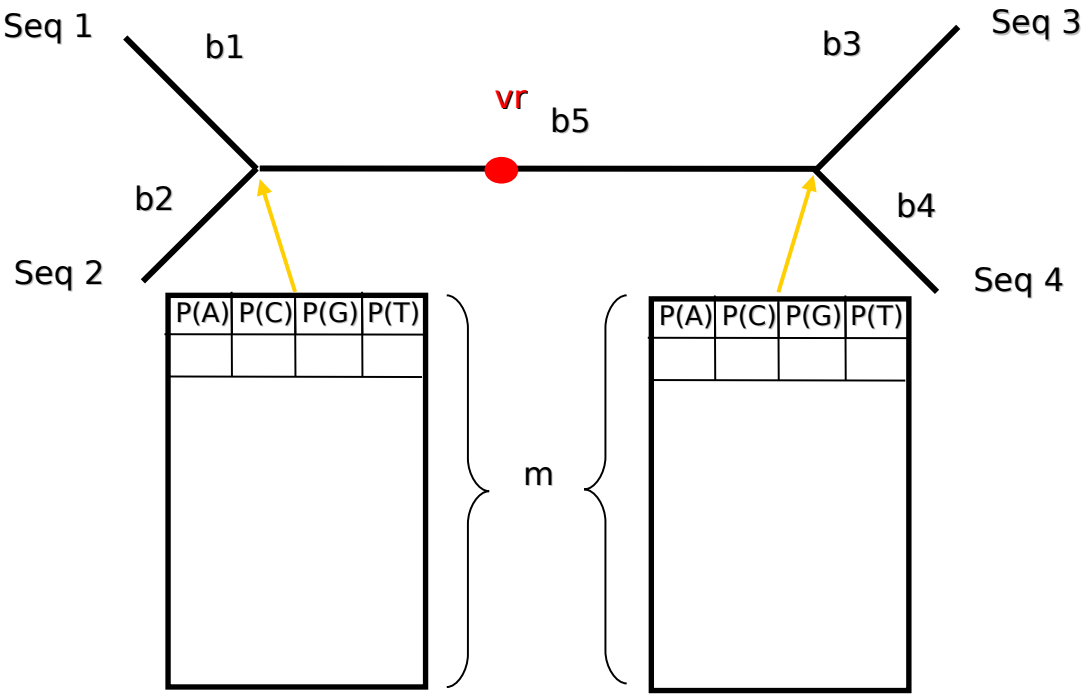
Maximum Likelihood



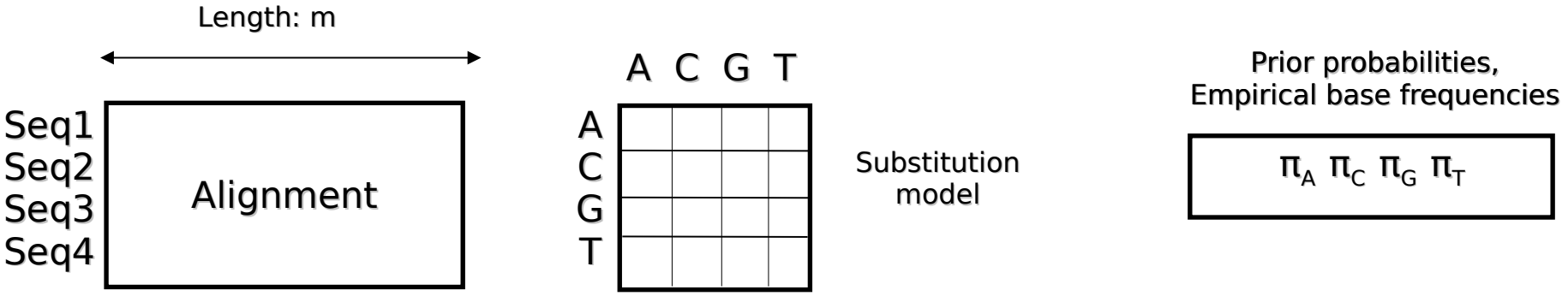


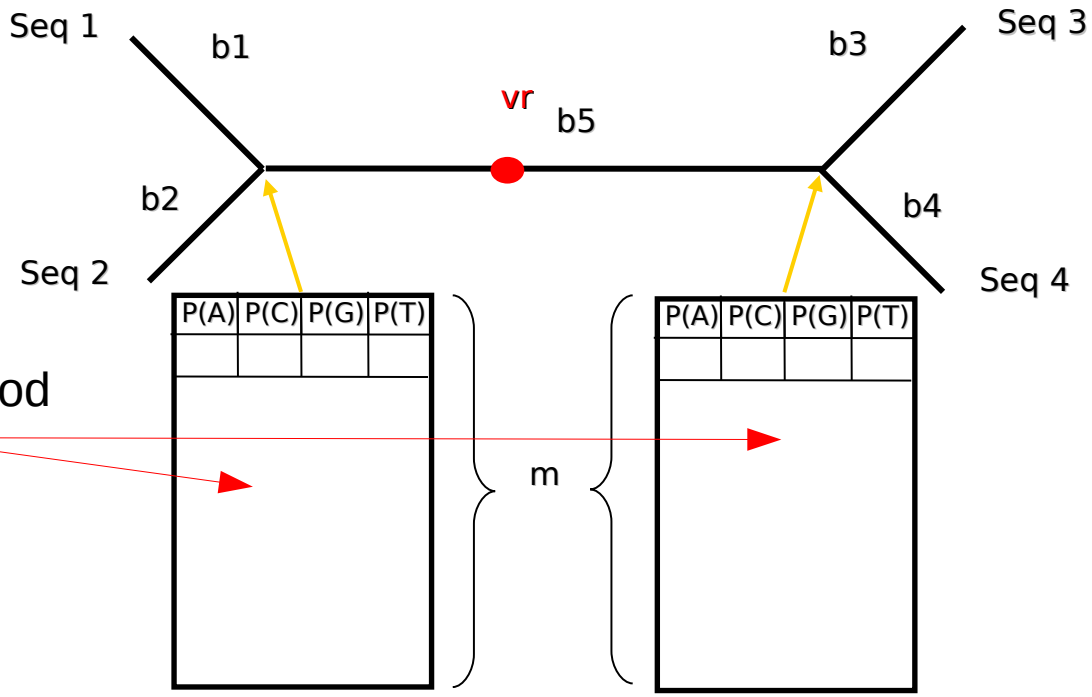
Maximum Likelihood





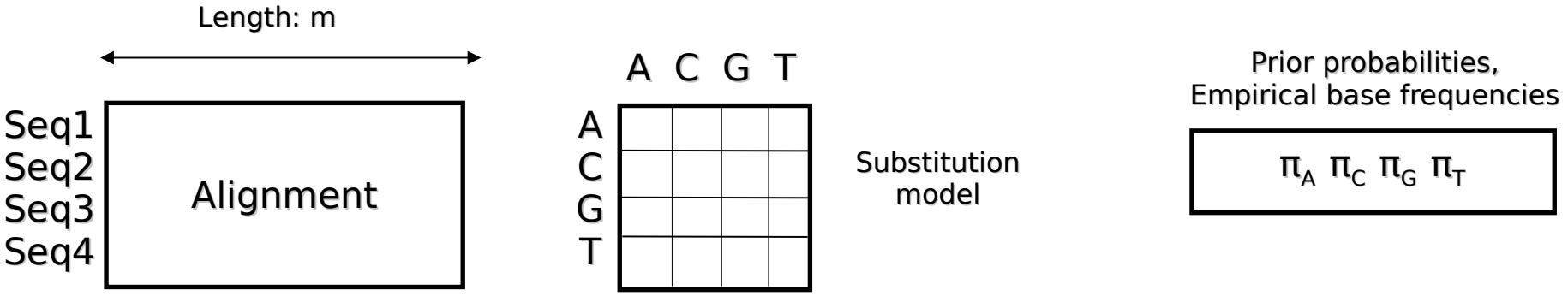
Maximum Likelihood

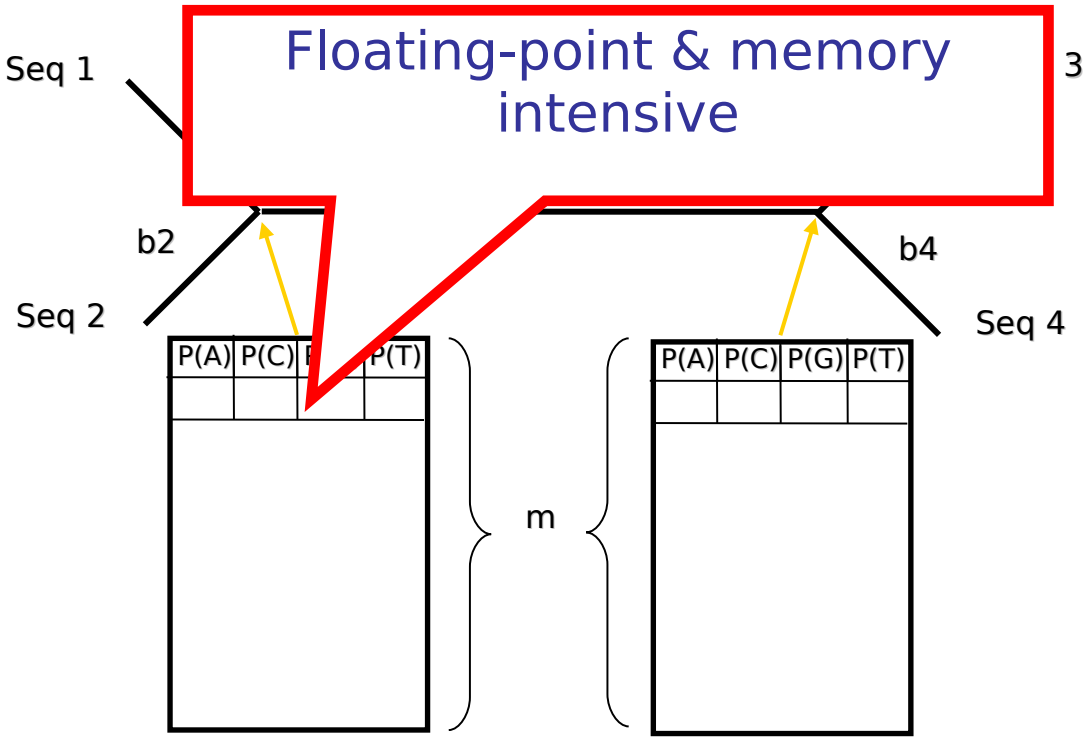




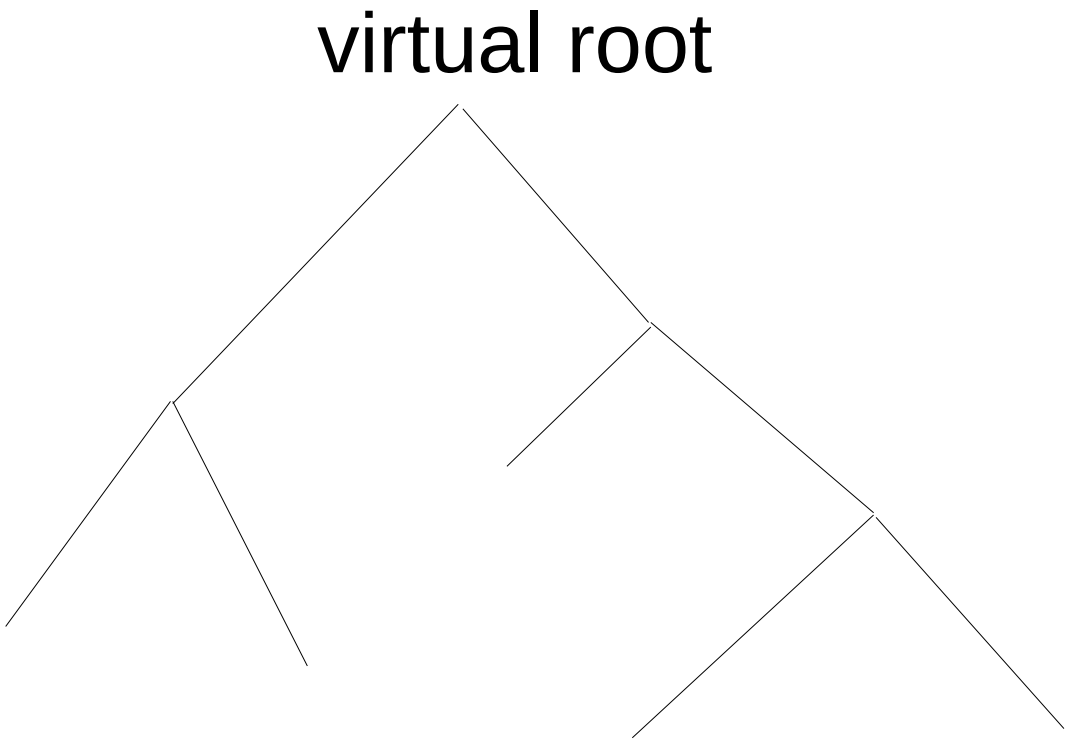
Conditional Likelihood Vectors

Maximum Likelihood

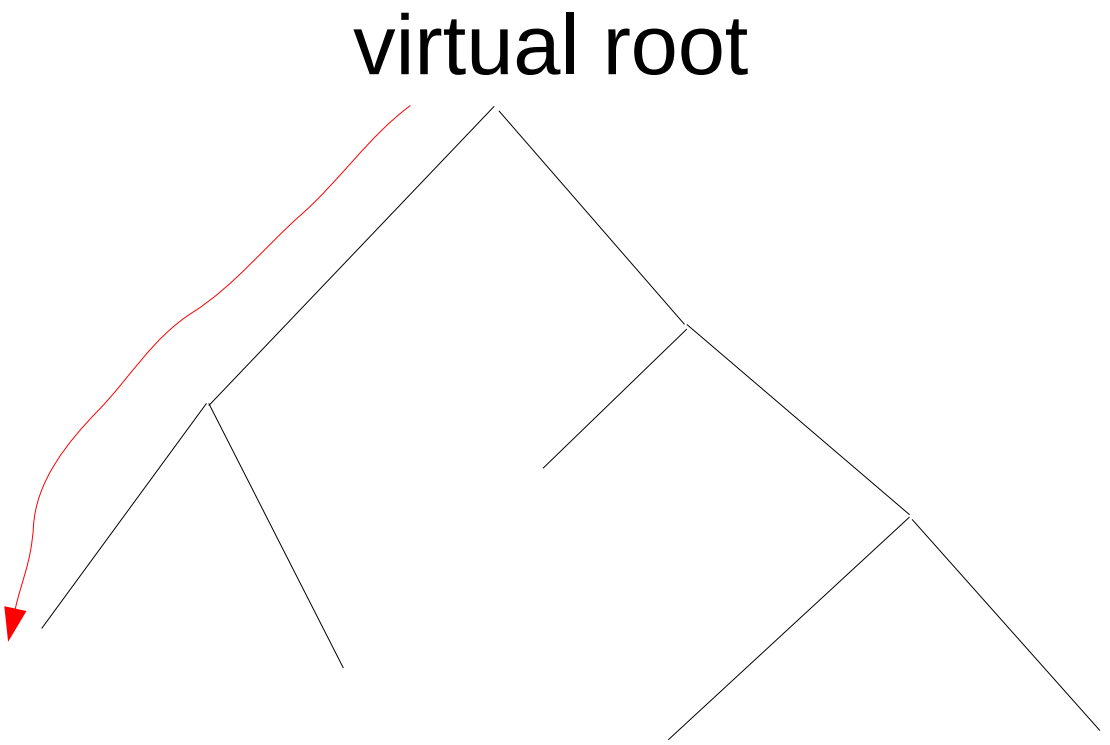




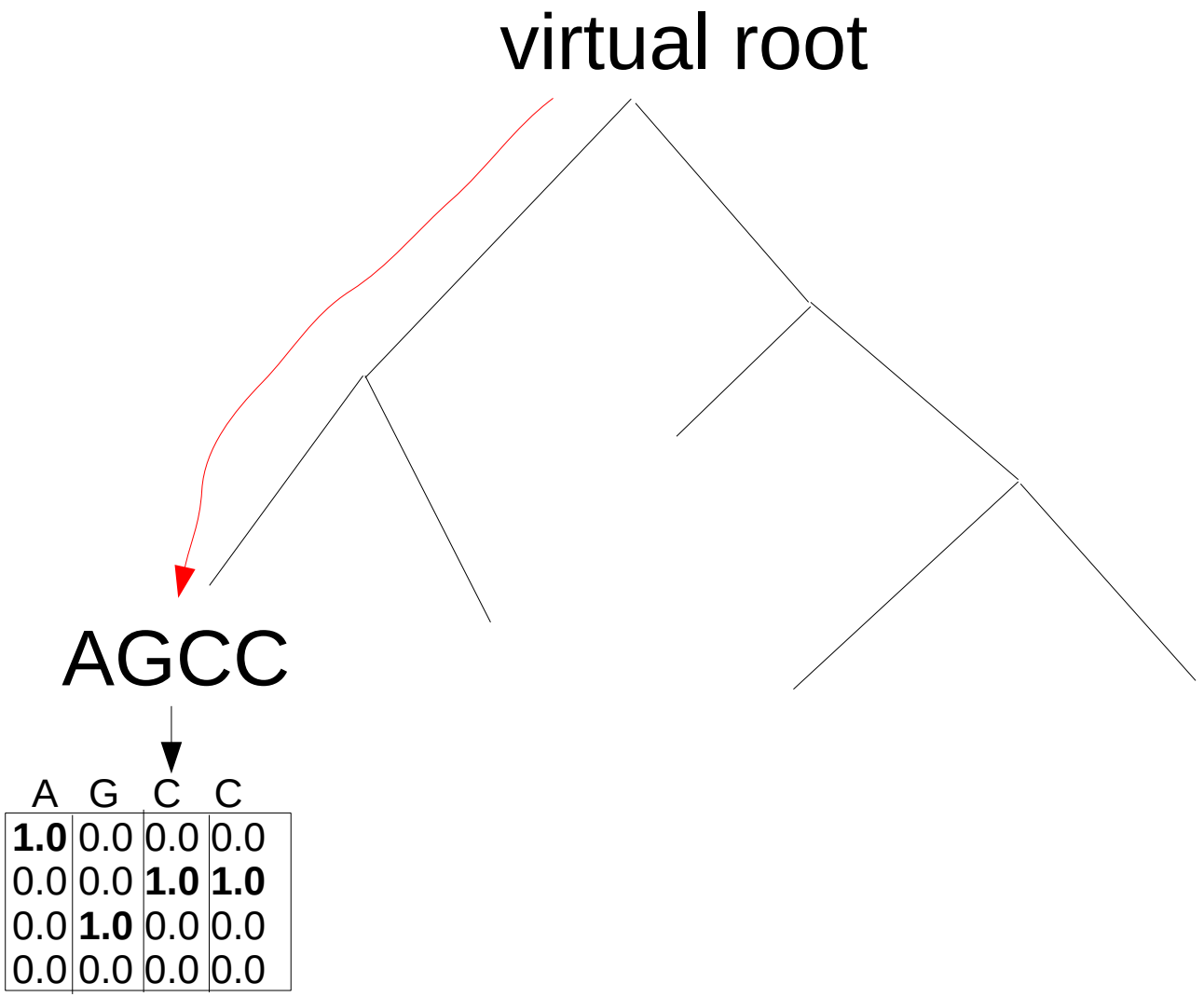
Post-order Traversal



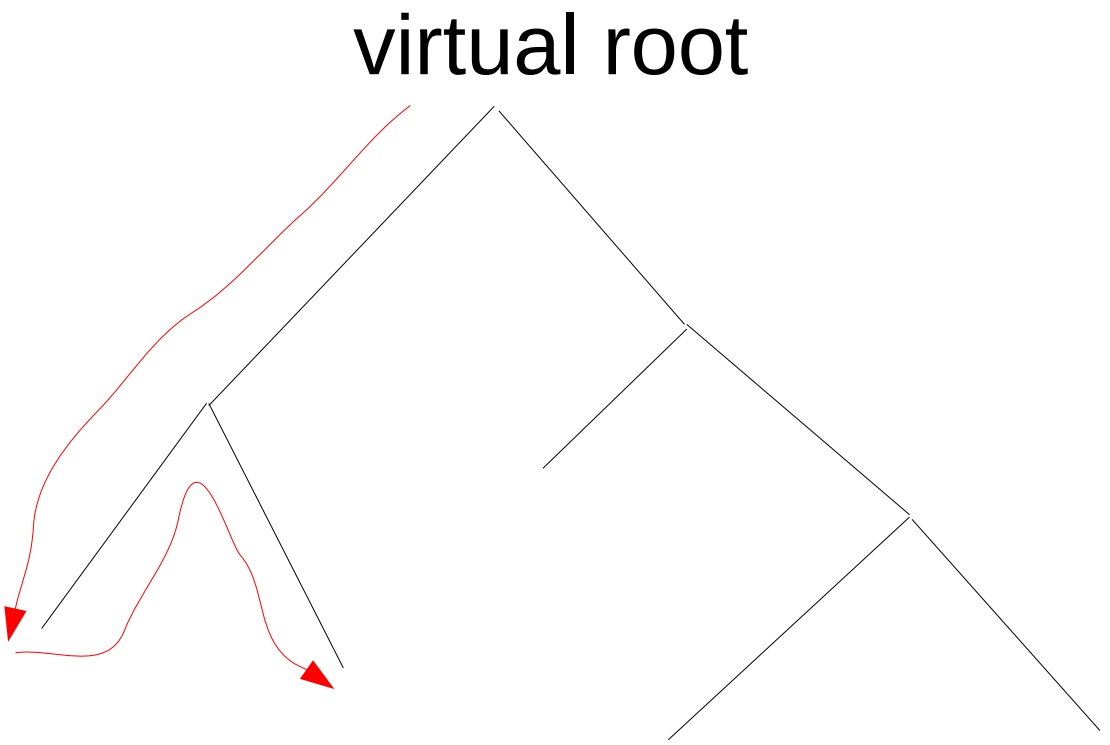
Post-order Traversal



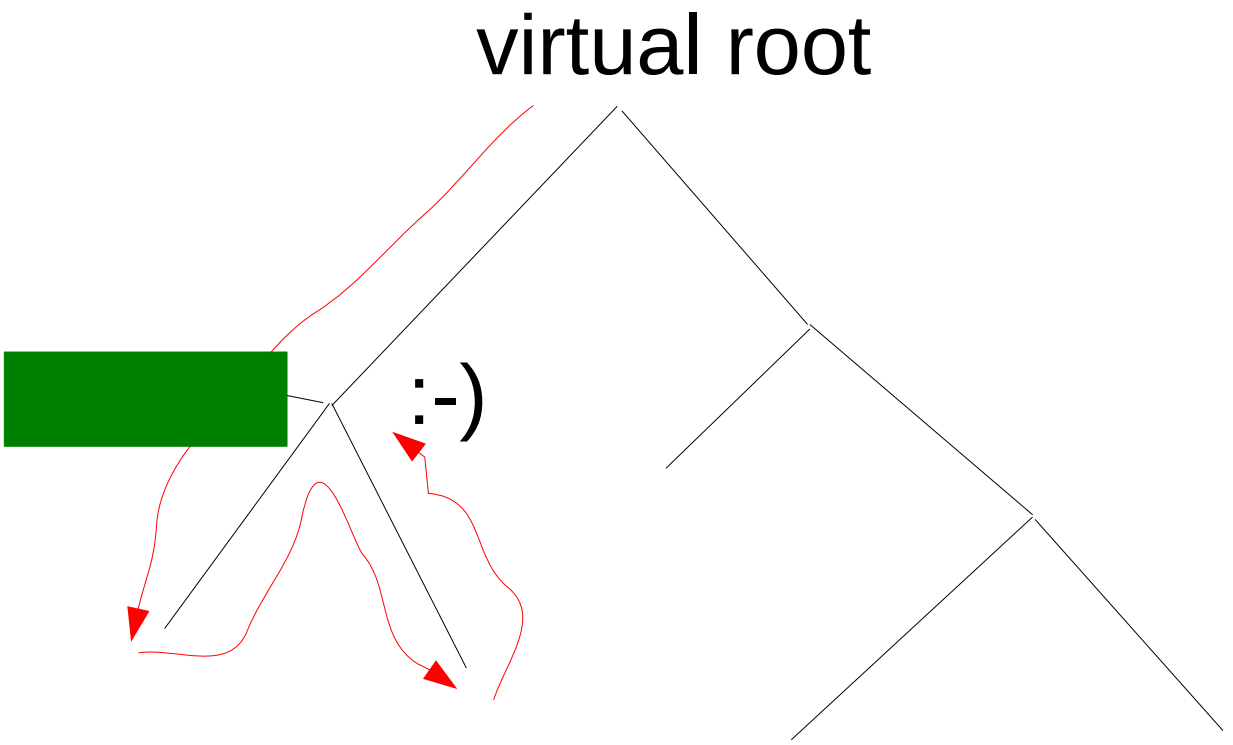
Post-order Traversal



Post-order Traversal

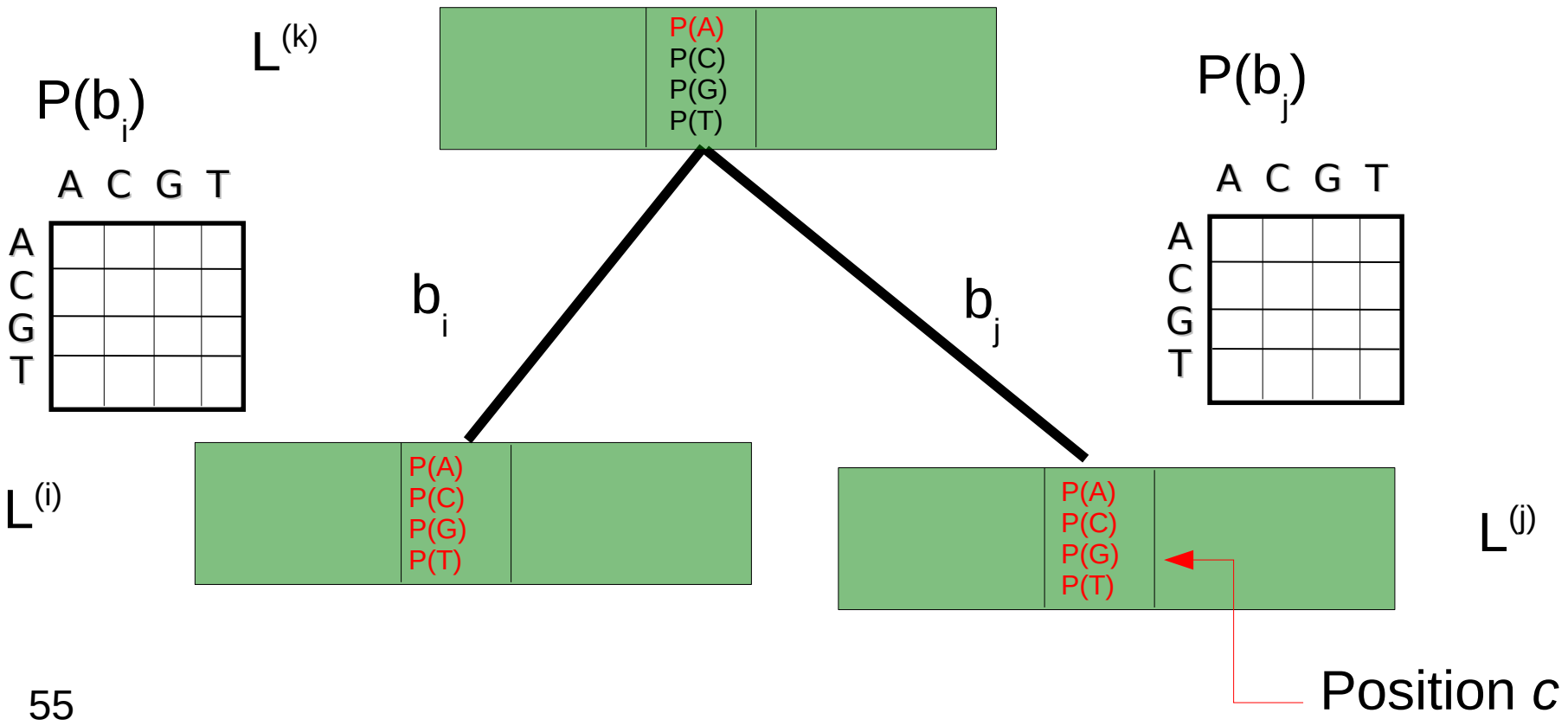


Post-order Traversal

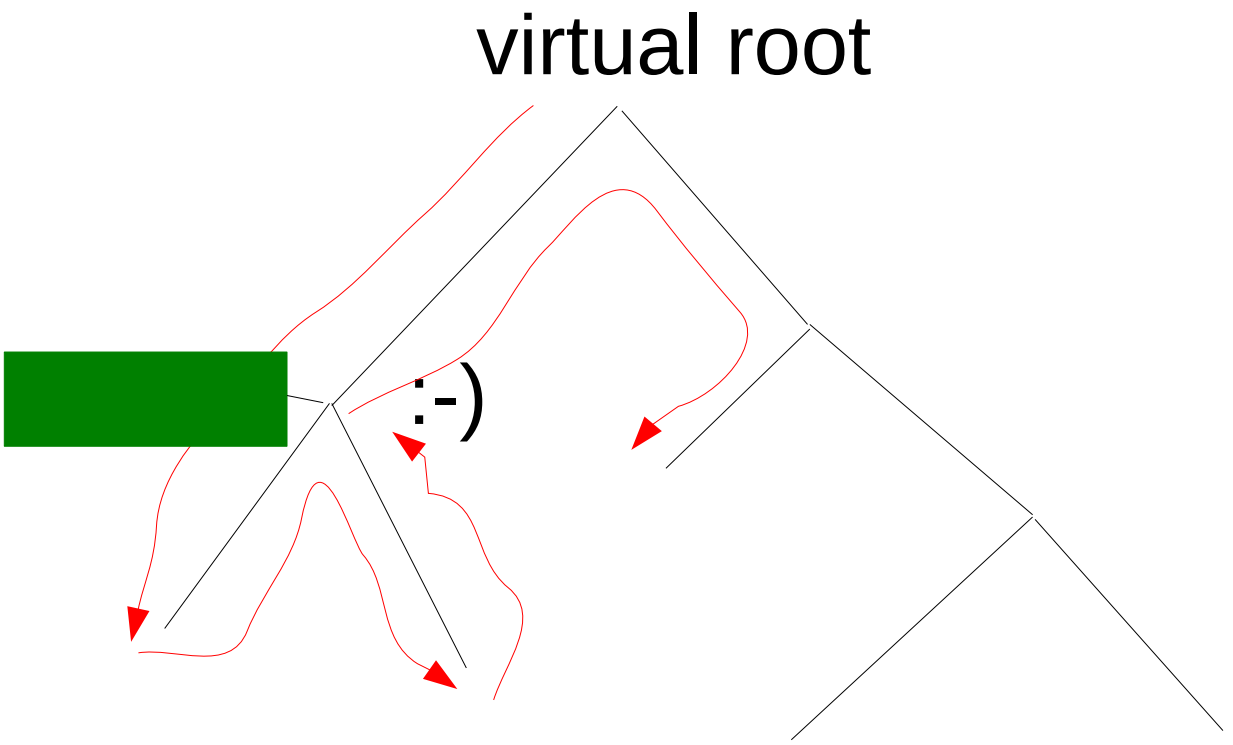


What happens when we compute this inner vector?

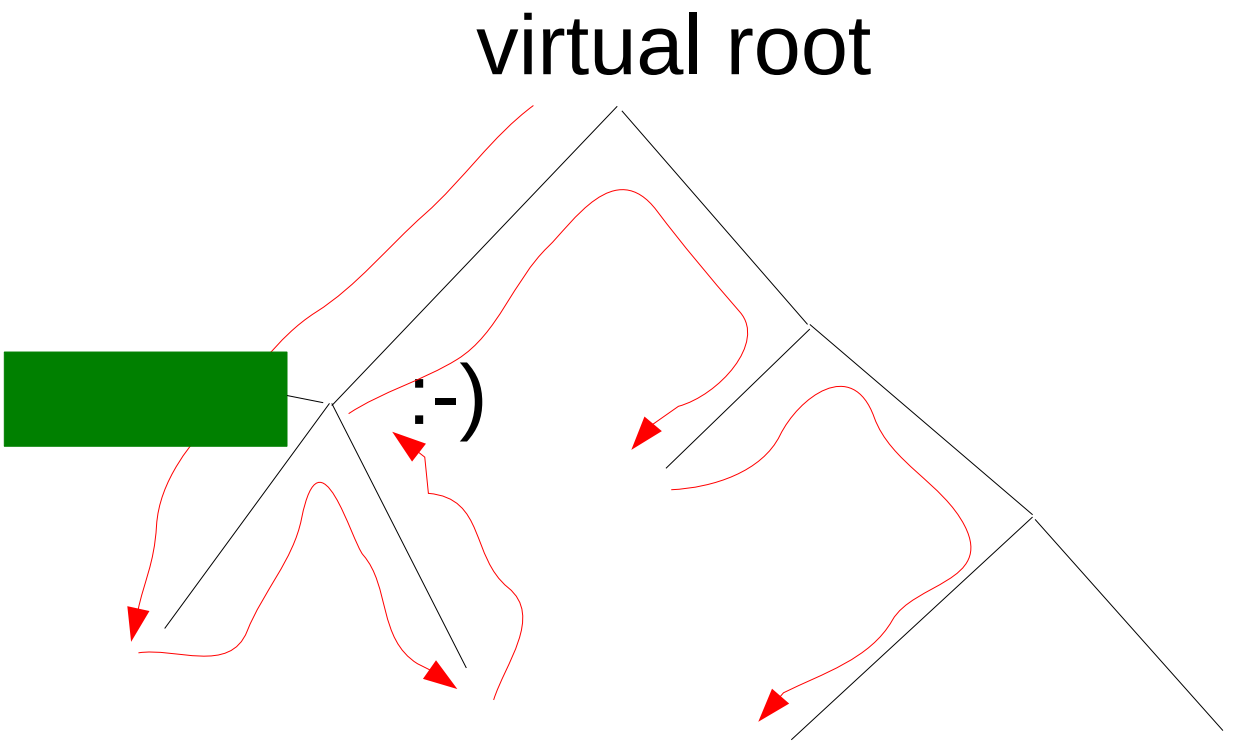
$$\vec{L}_A^{(k)}(c) = \left(\sum_{S=A}^{T} P_{AS}(b_i)\vec{L}_S^{(i)}(c)\right)\left(\sum_{S=A}^{T} P_{AS}(b_j)\vec{L}_S^{(j)}(c)\right)$$



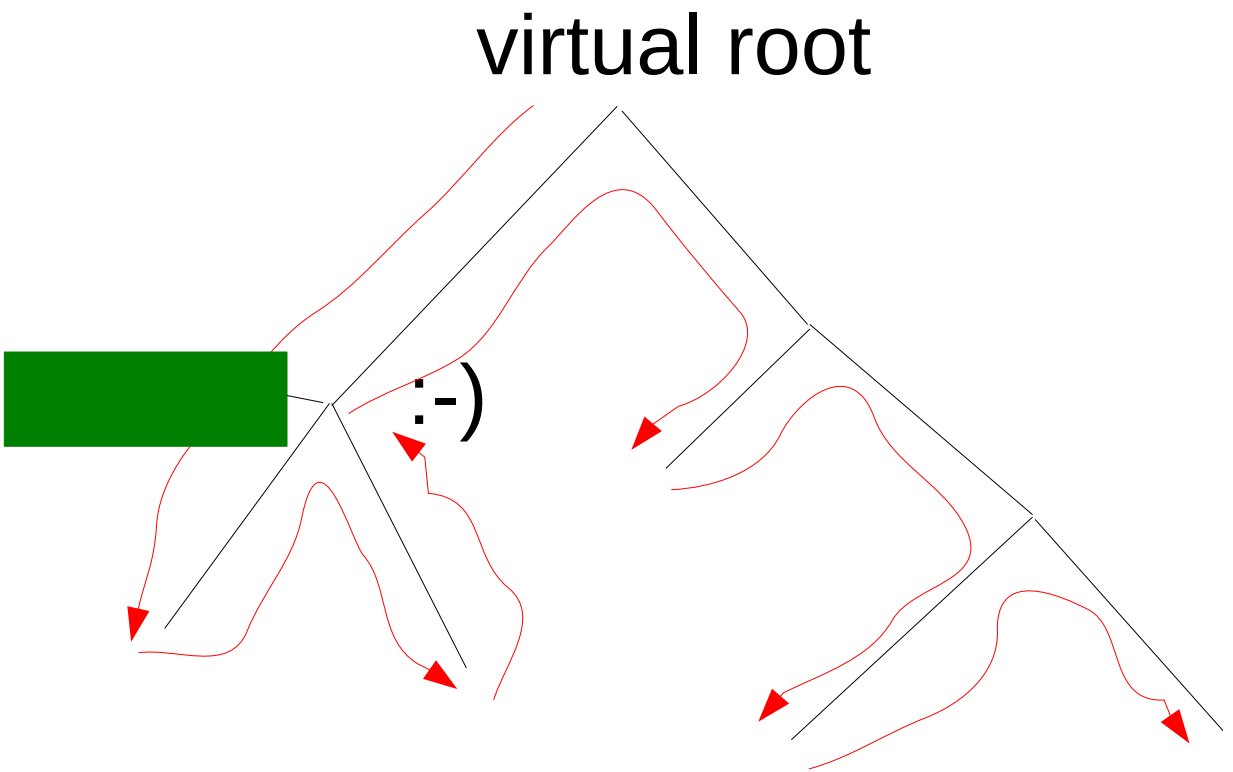
Post-order Traversal



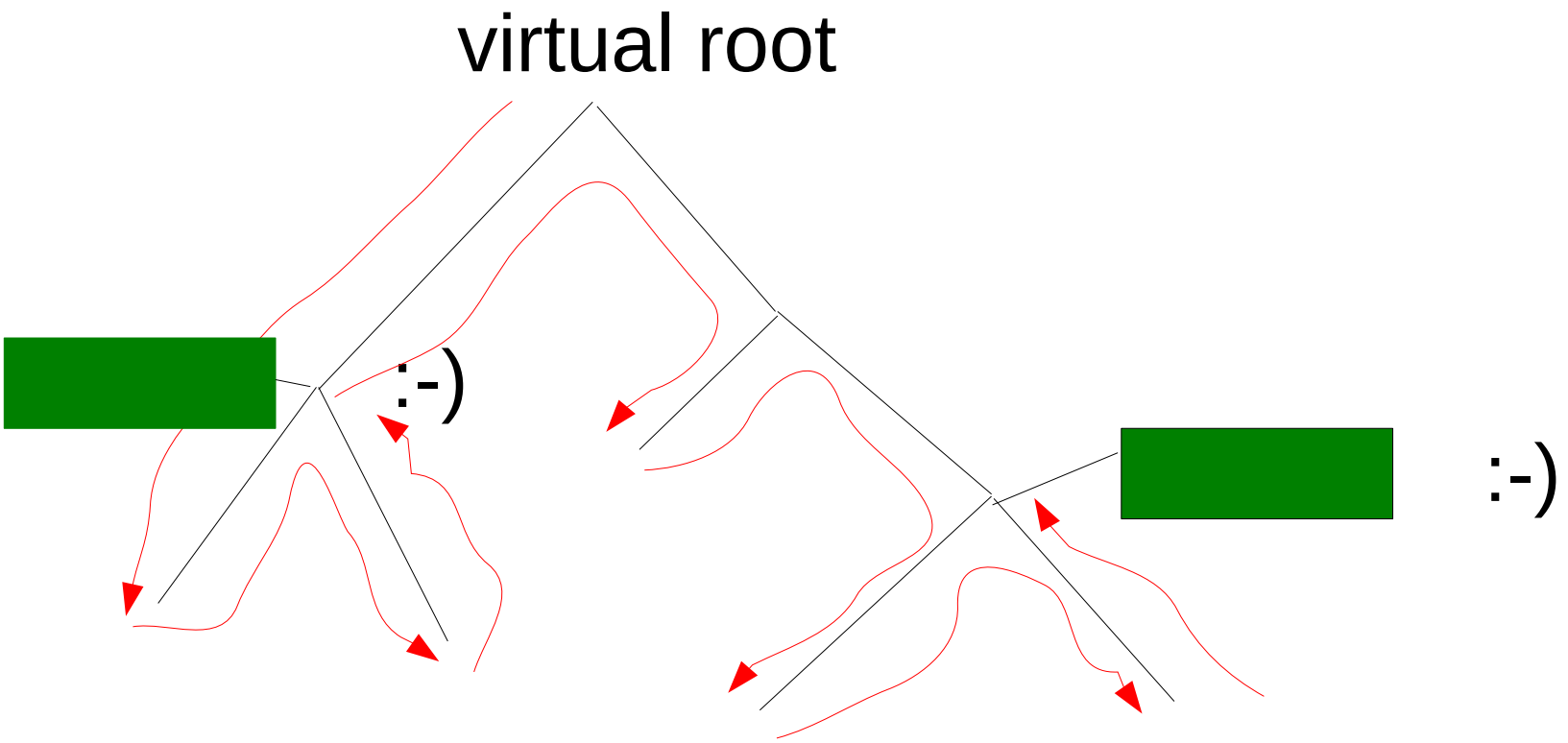
Post-order Traversal



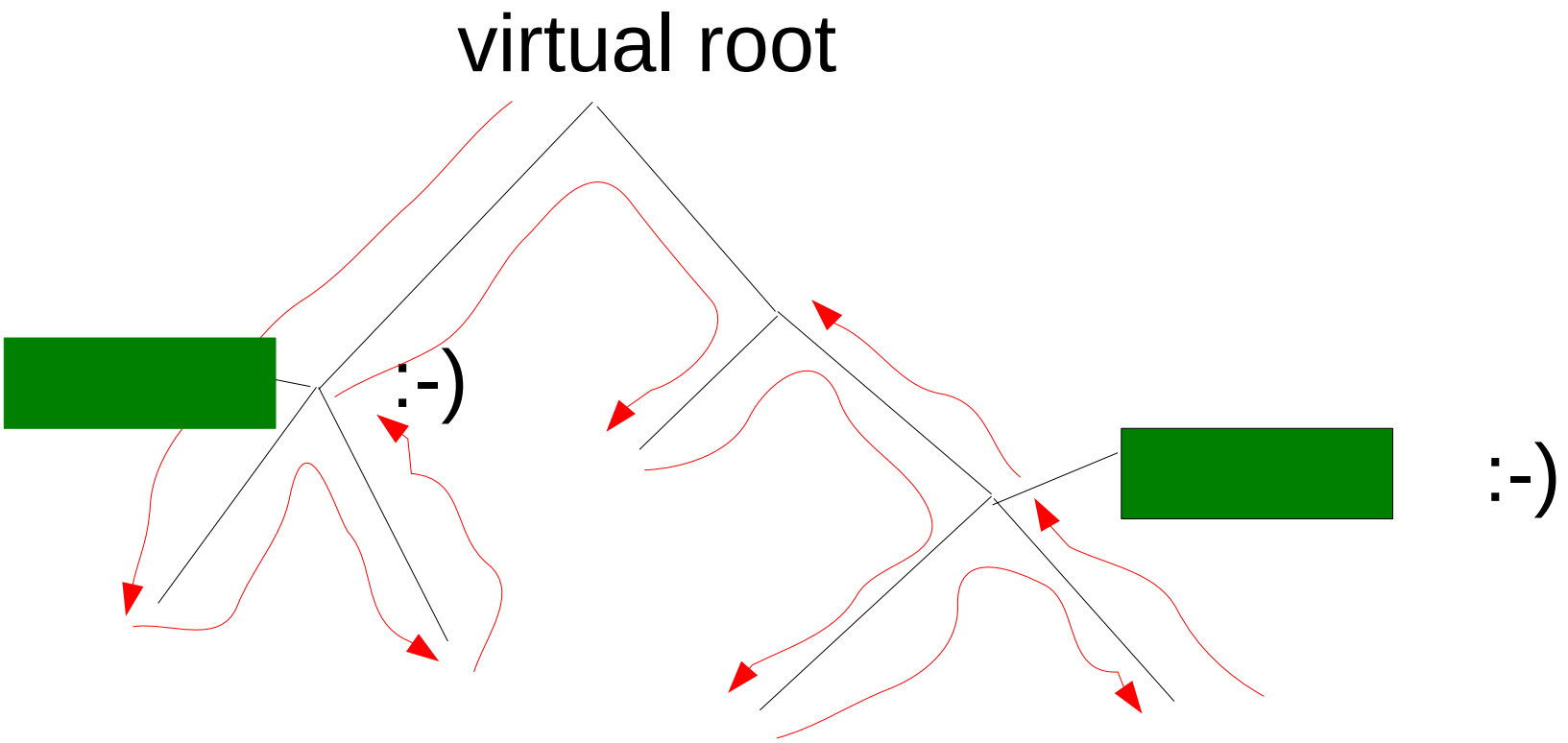
Post-order Traversal



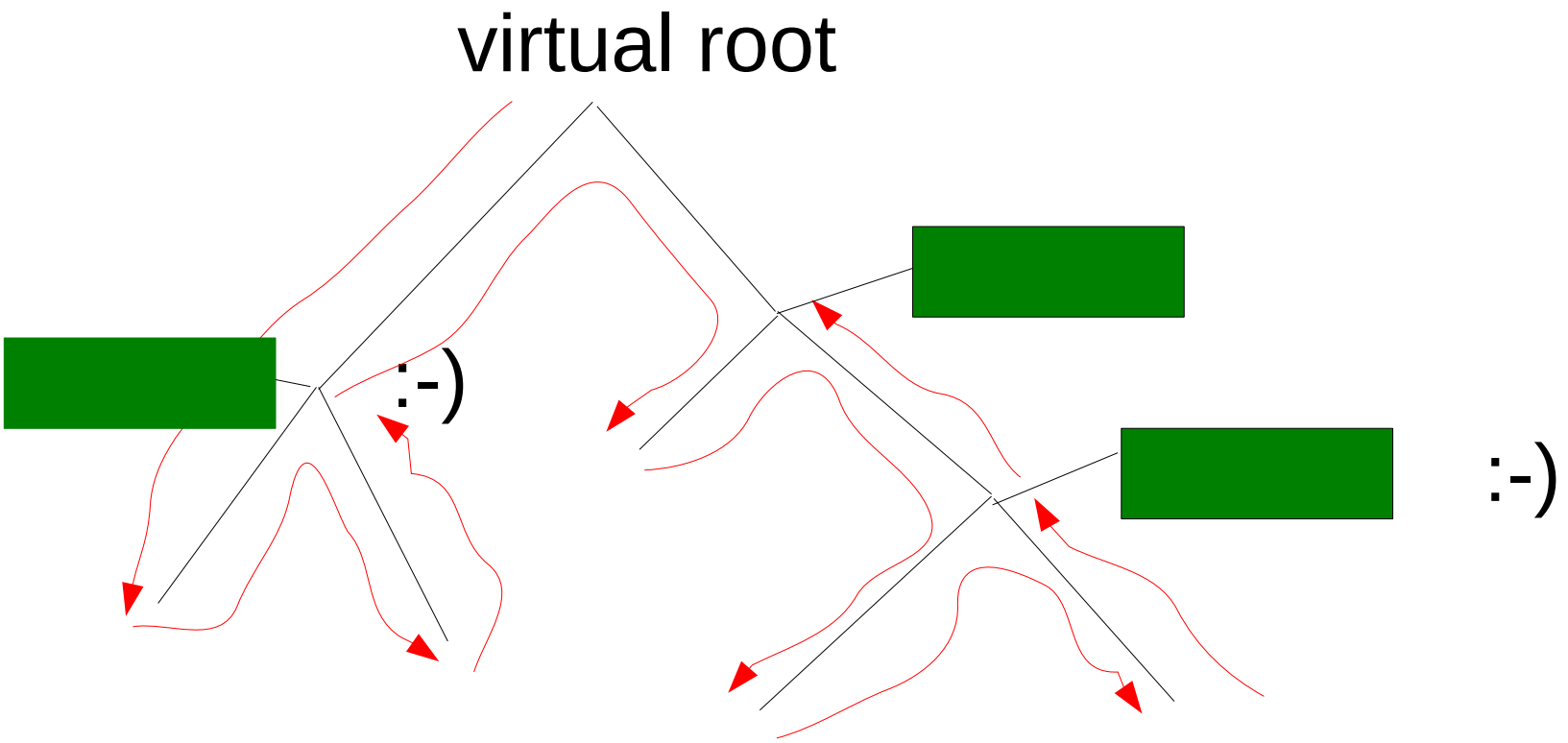
Post-order Traversal



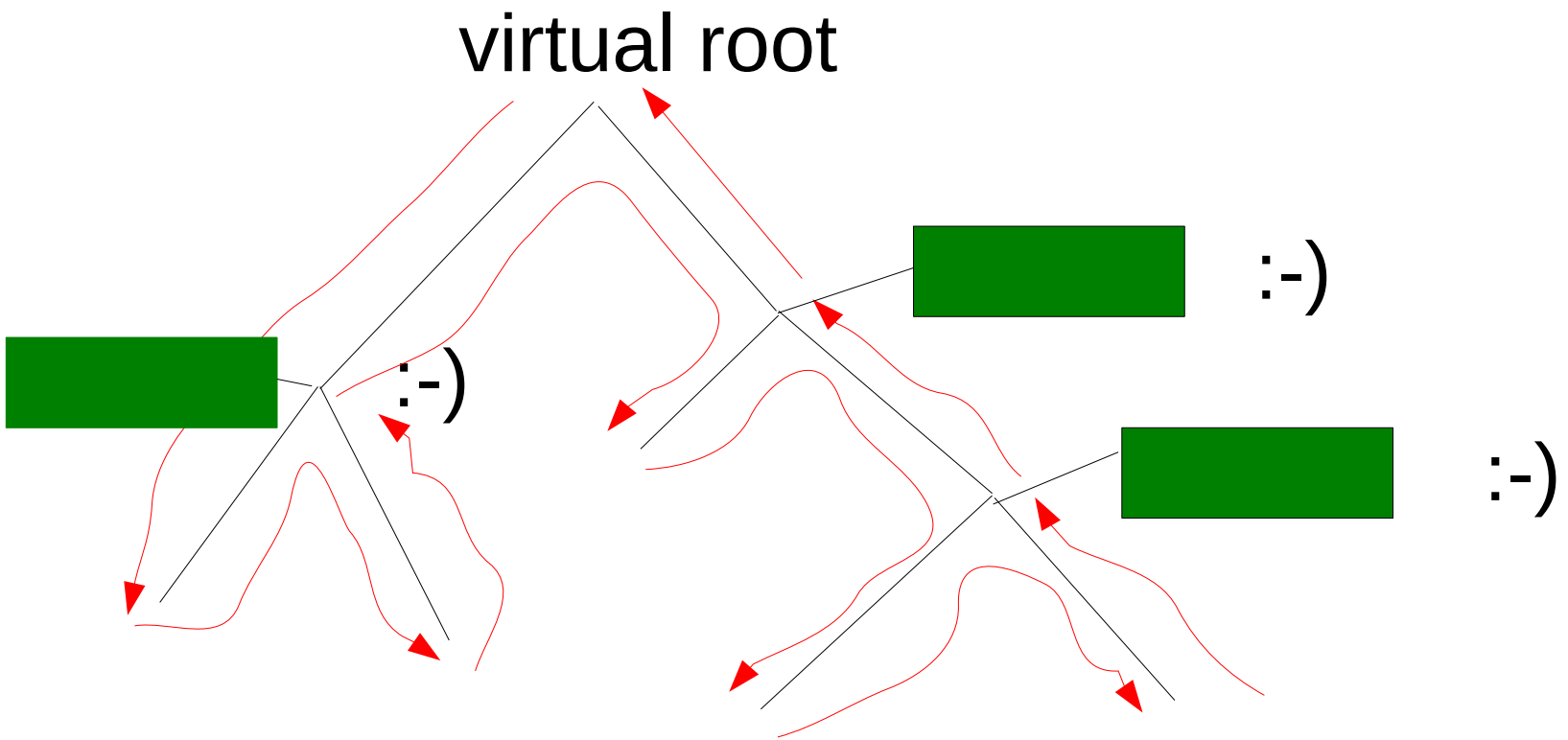
Post-order Traversal



Post-order Traversal

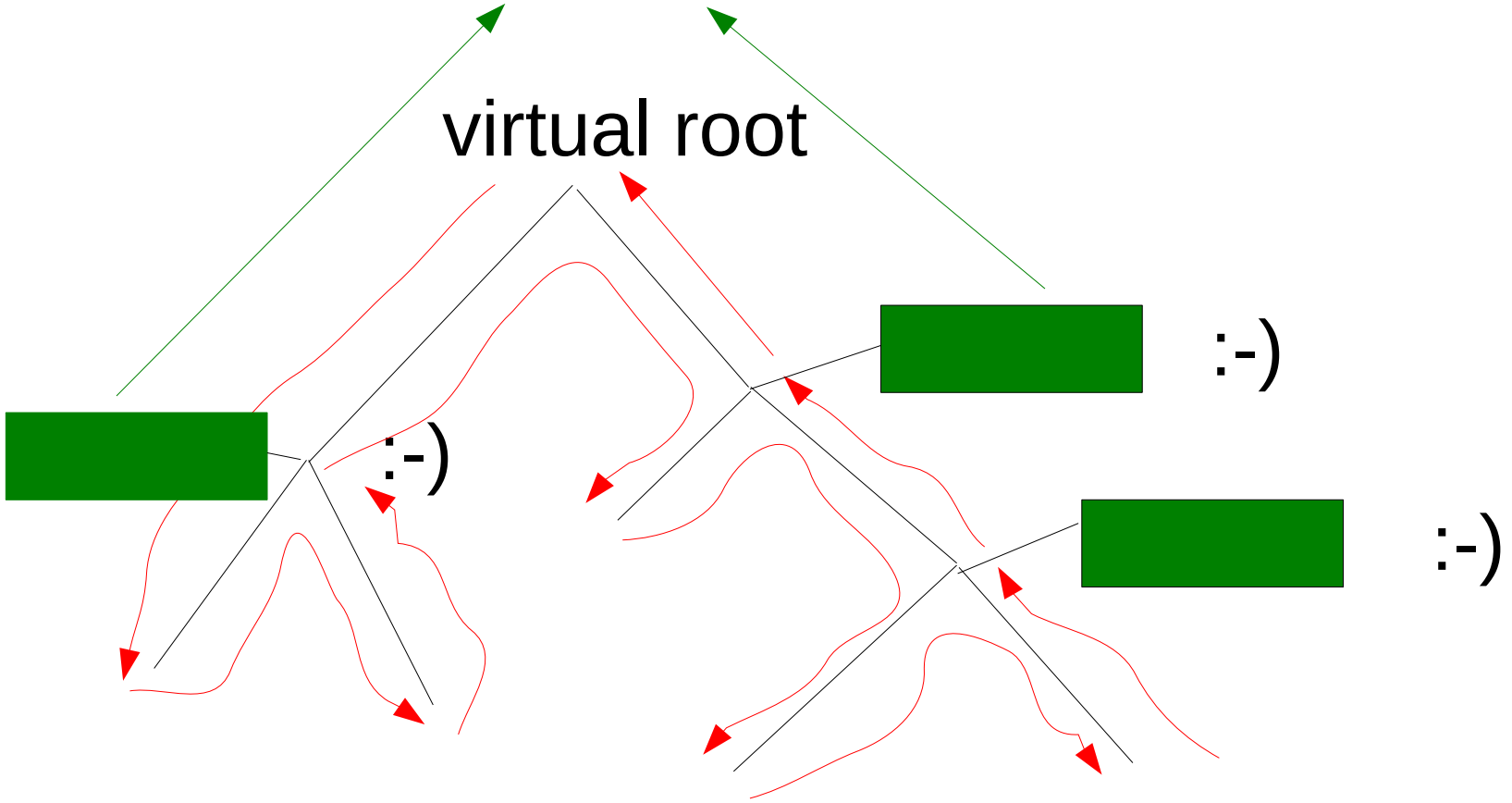


Post-order Traversal



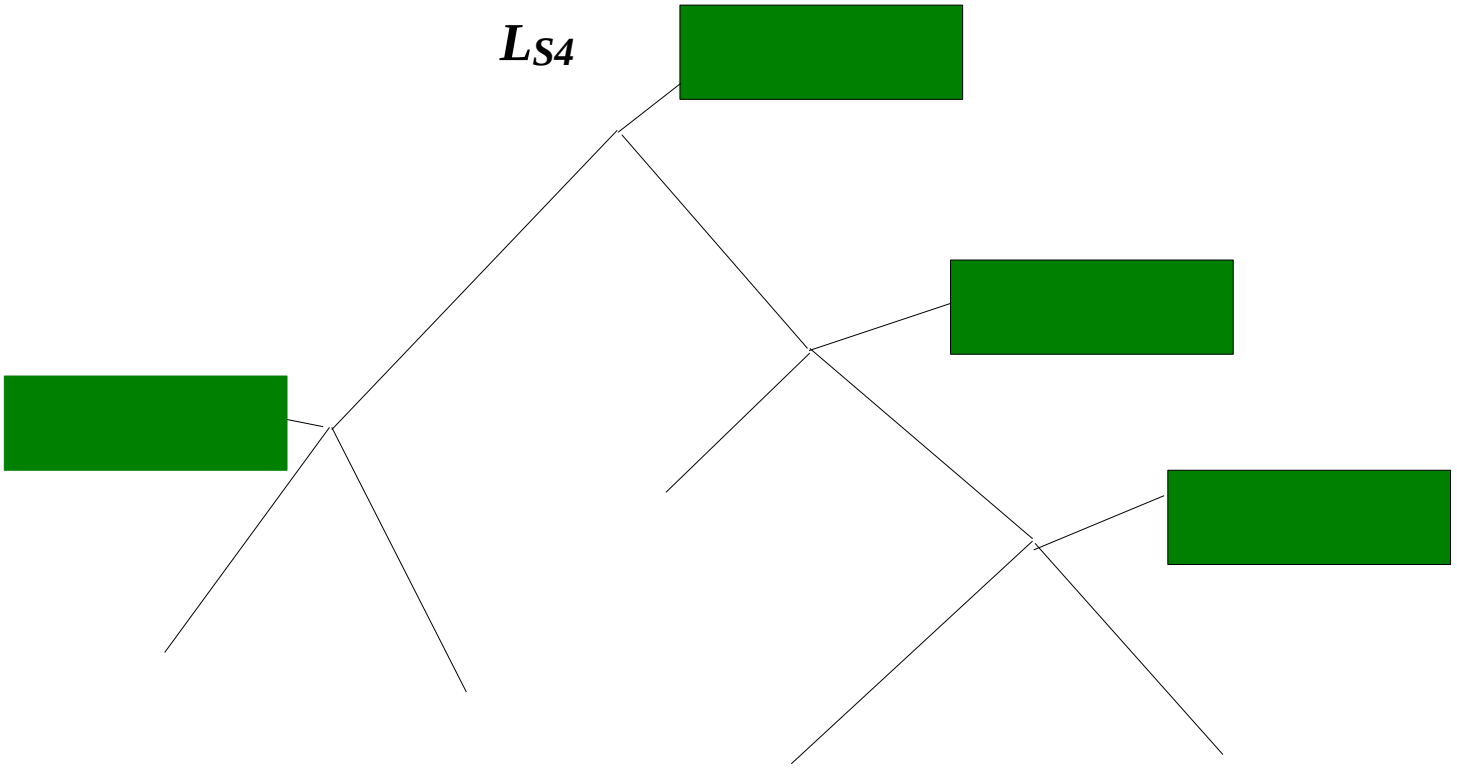
Post-order Traversal

Overall likelihood: sum over logarithms of per-site likelihoods

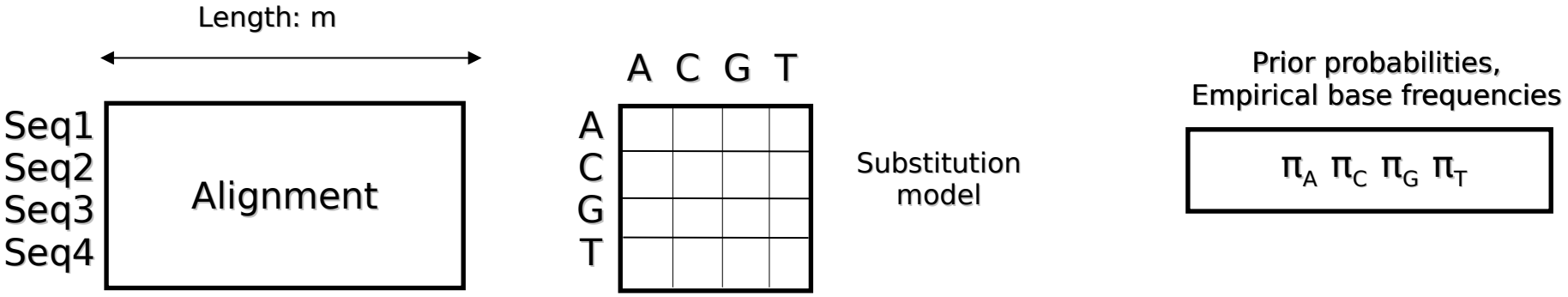


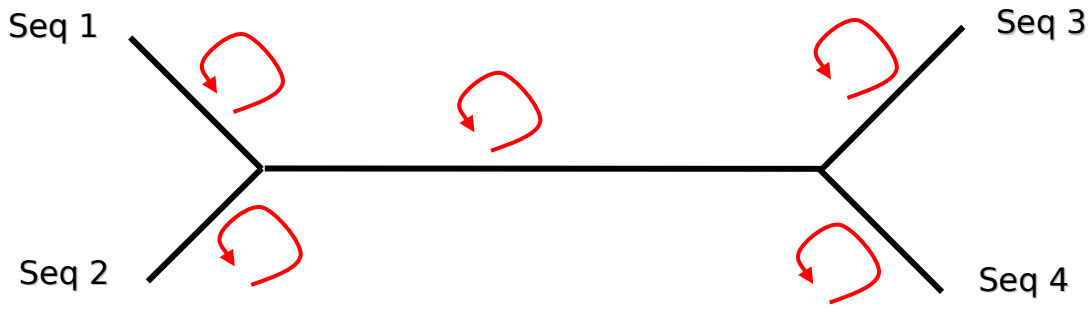
Post-order Traversal

$$L = \sum_{S_4=A}^{T} \pi_{S_4} L_{S_4}$$



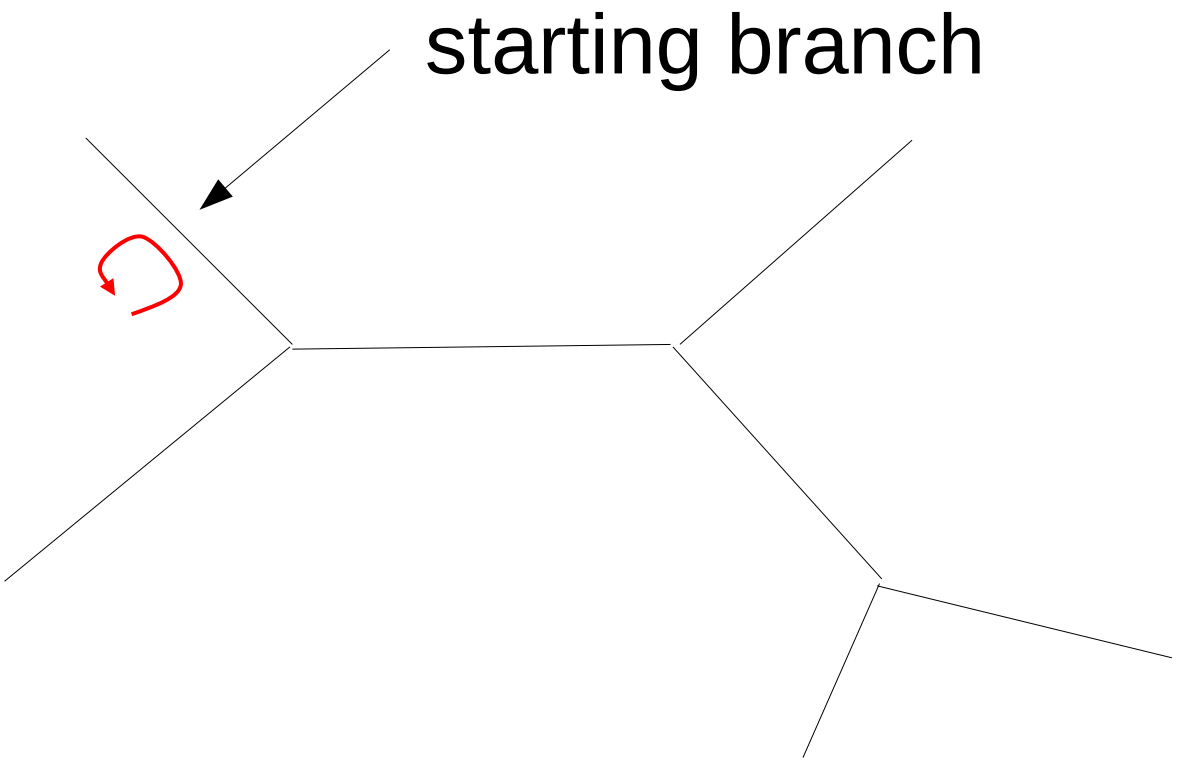
Maximum Likelihood



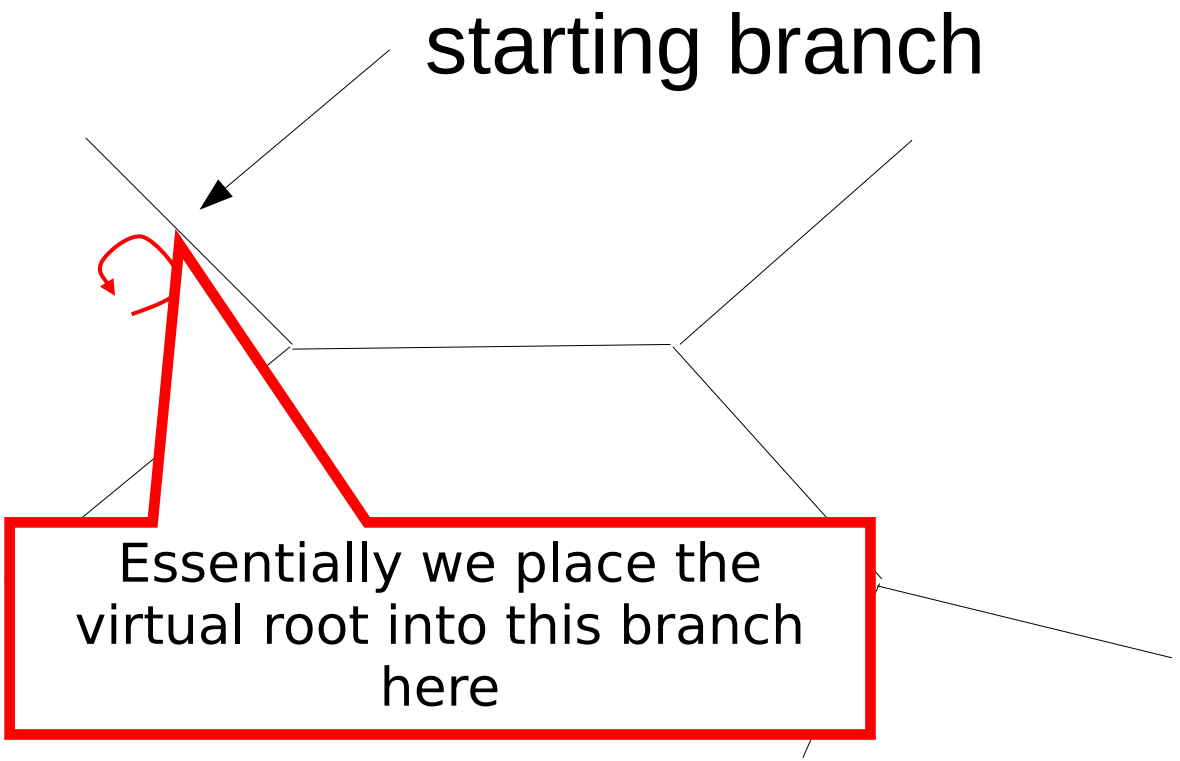


optimize branch lengths

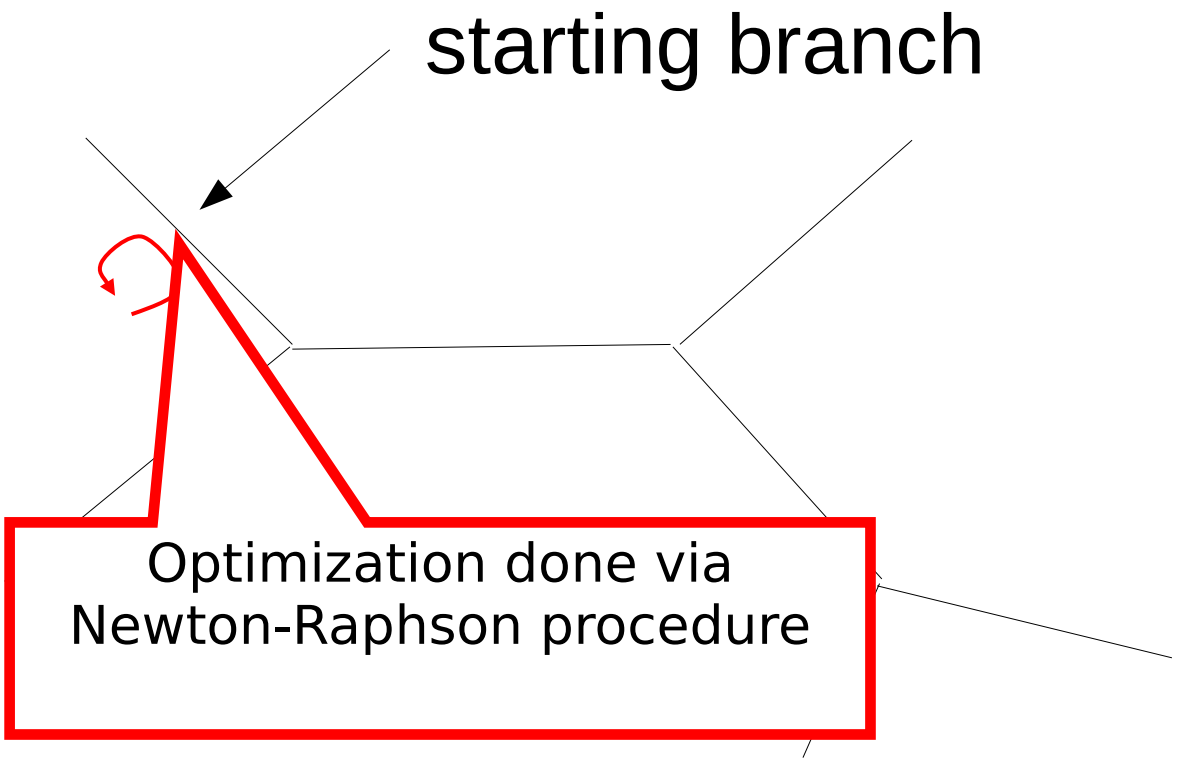
Branch Length Optimization



Branch Length Optimization



Branch Length Optimization



Newton Rapshon

- We want to find the branch length *b* that maximizes the likelihood *L(b)* of the tree
- For this, we want to know where the *first* derivative of *L(b)* is *0*
- To achieve this numerically we use the Newton-Raphson procedure for root finding deploying the first and second derivative of the likelihood *L'(b)* and *L''(b)*
- Note that, the likelihood only depends on branch *b*, all other model parameters (*Q* matrix, base frequencies, tree topology) remain fixed

Derivatives of *L(b)*

- To compute the derivatives of *L(b)*, we essentially need to be able to compute the derivatives of *P(b)* since the rest is just sums and does not depend on *b*
- Recall

  $P(b) = e^{Qb} = Ue^{\Lambda b}U^{-1}$

- thus

  $(P(b))' = U\Lambda e^{\Lambda b}U^{-1}$

- and

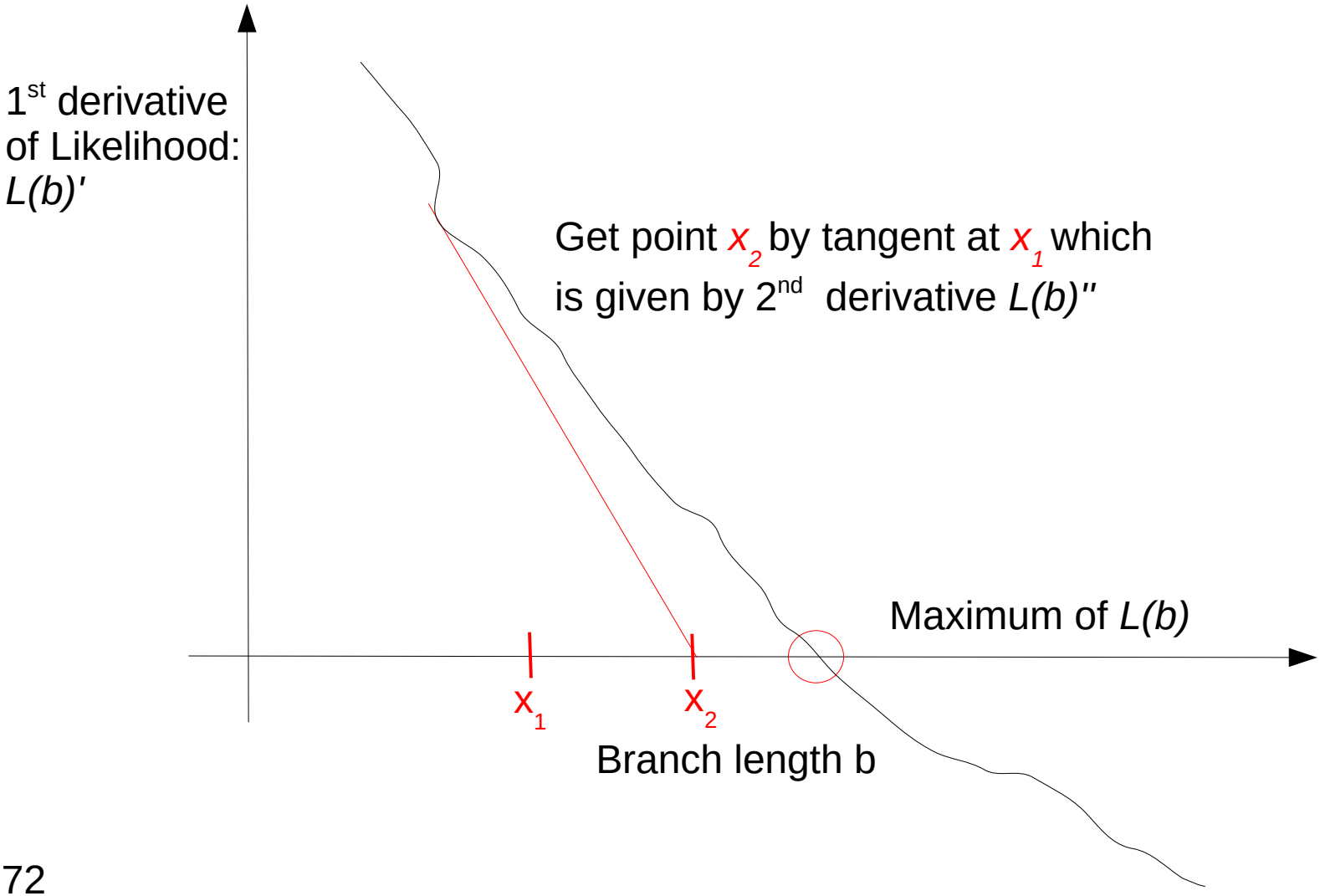
  $(P(b))'' = U\Lambda^2 e^{\Lambda b}U^{-1}$

- In practice we compute the derivatives of the log likelihood *log(L(b))*, but it is essentially the same (see next slide)

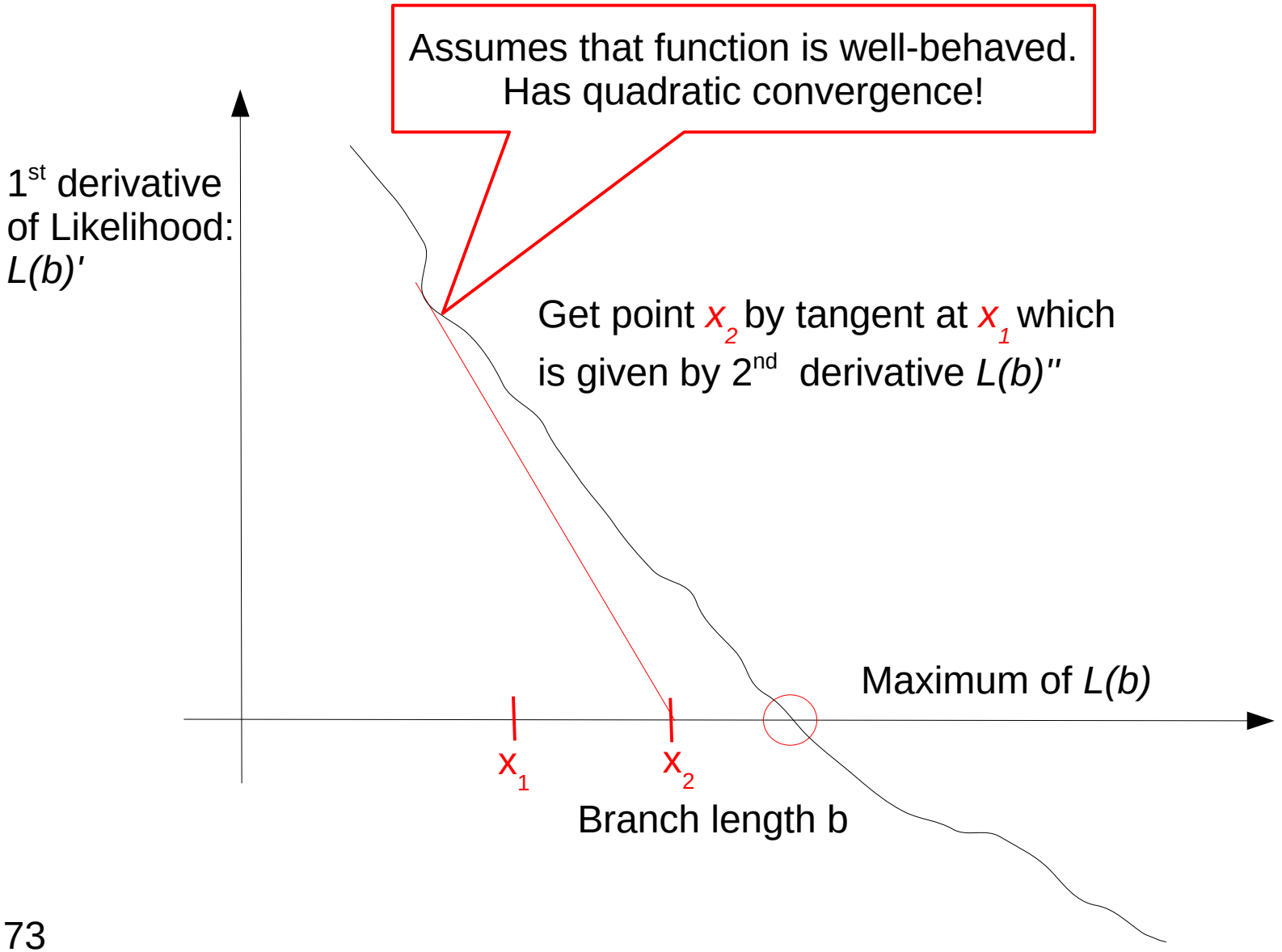
Derivatives of *log(L(b))*

- 1st derivative: $L(b)'/L(b)$
- *2nd derivative:* $(L(b)\,L(b)'' - (L(b)'')^2)\,/\,L(b)^2$

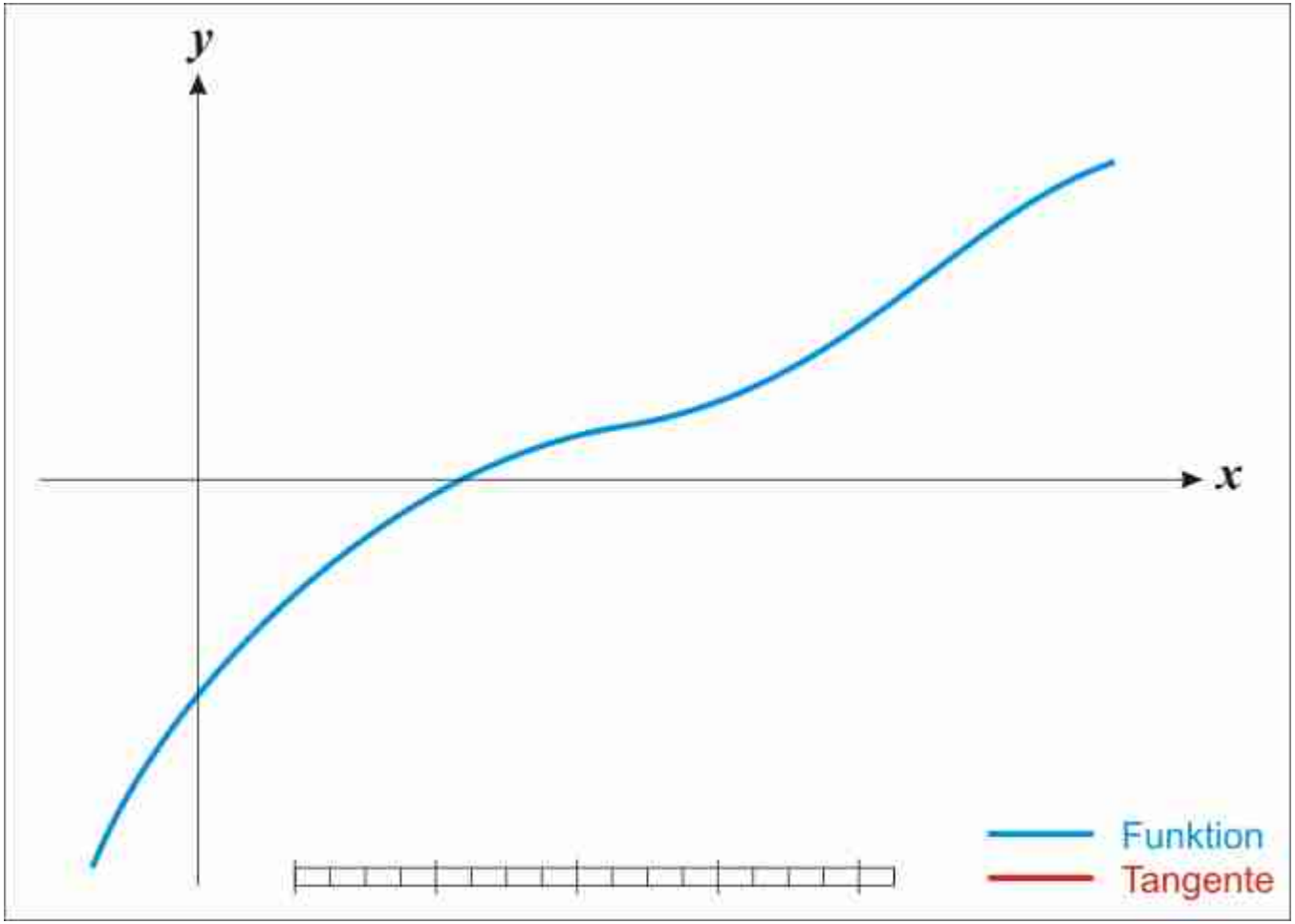
Newton Raphson



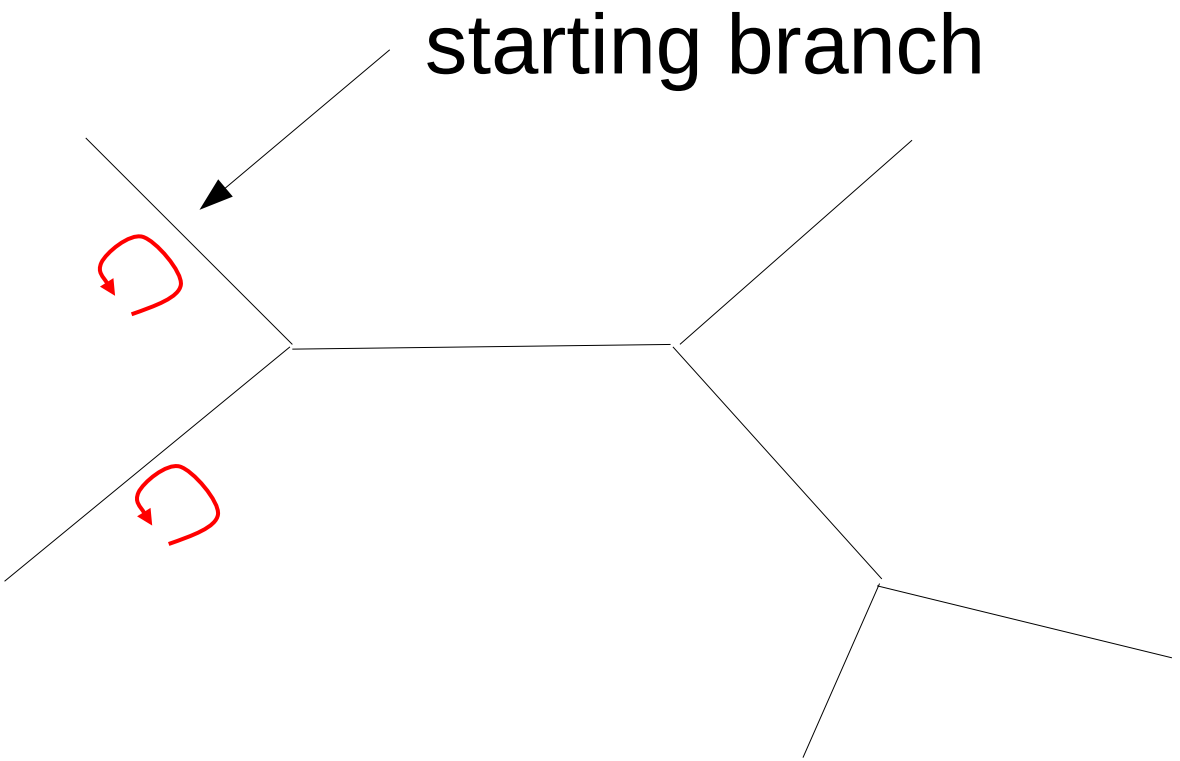
Newton Raphson



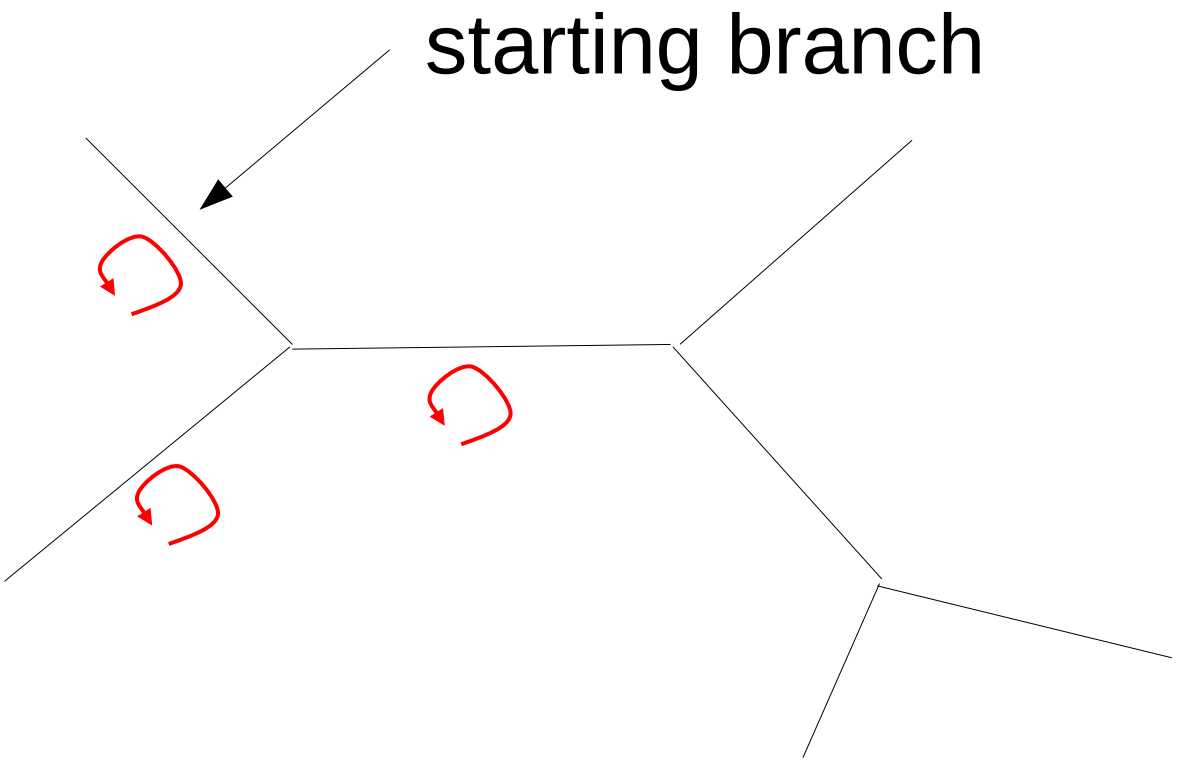
An animation



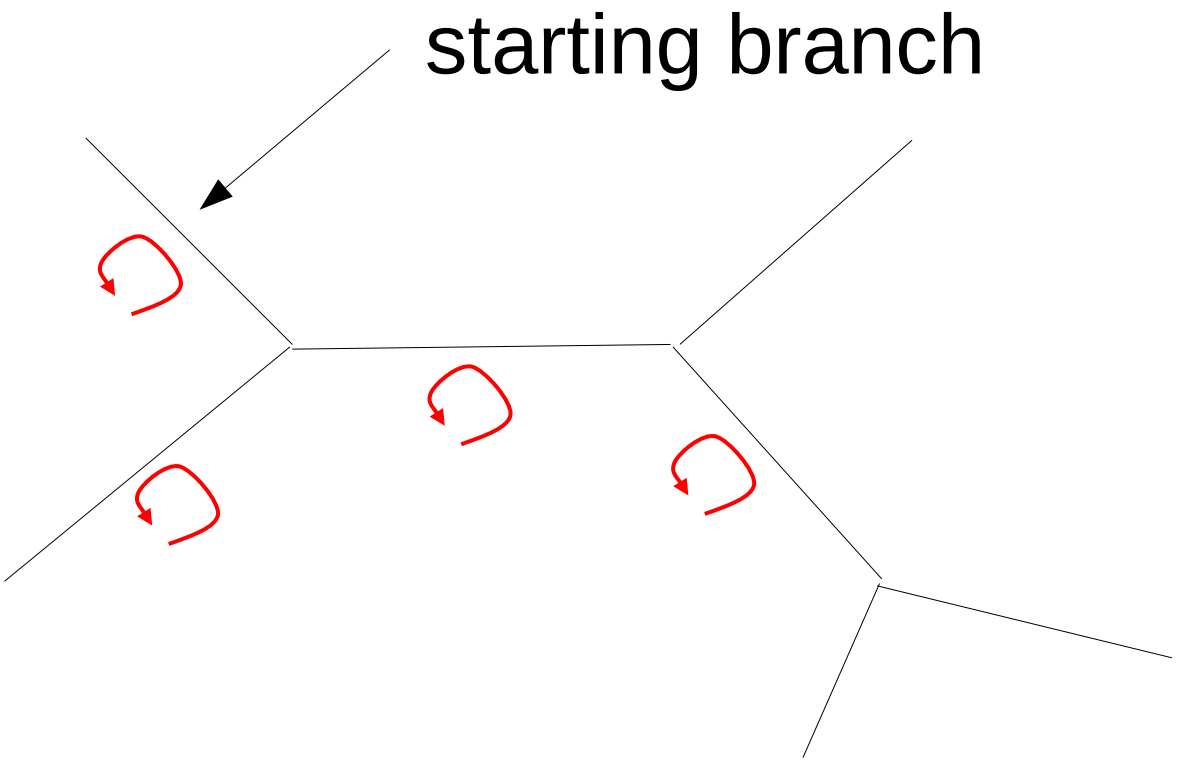
Branch Length Optimization



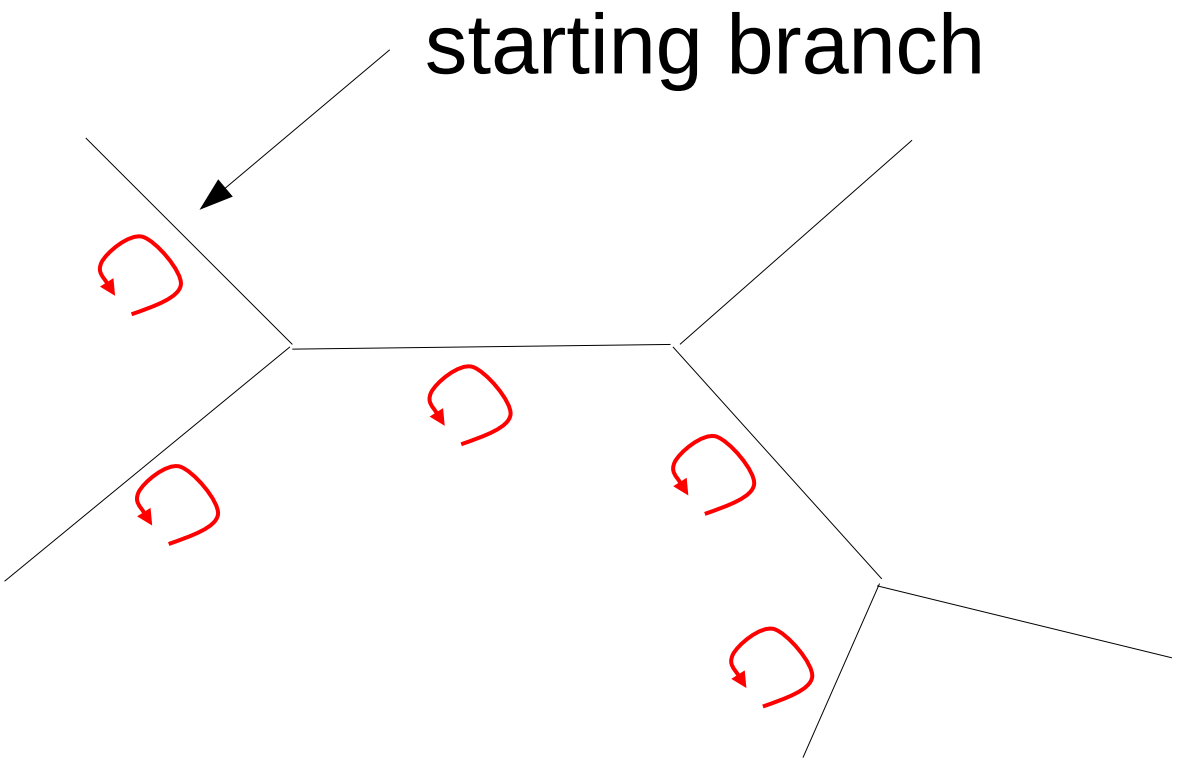
Branch Length Optimization



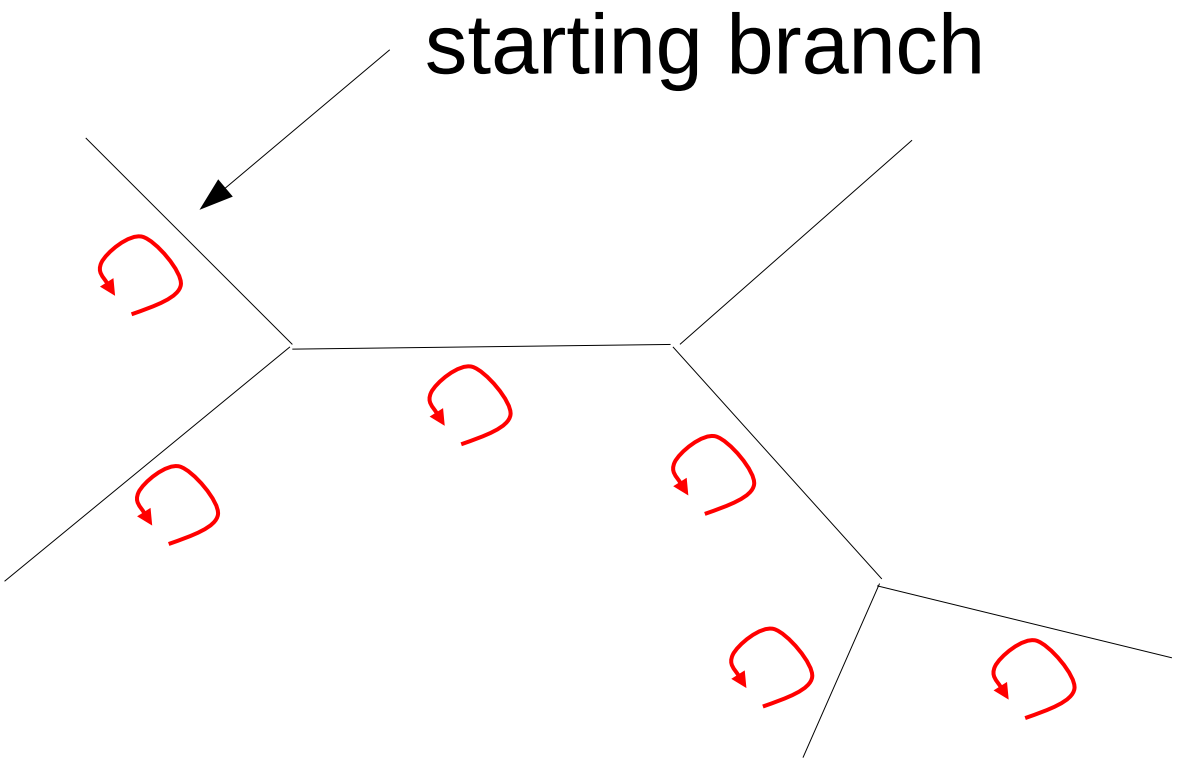
Branch Length Optimization



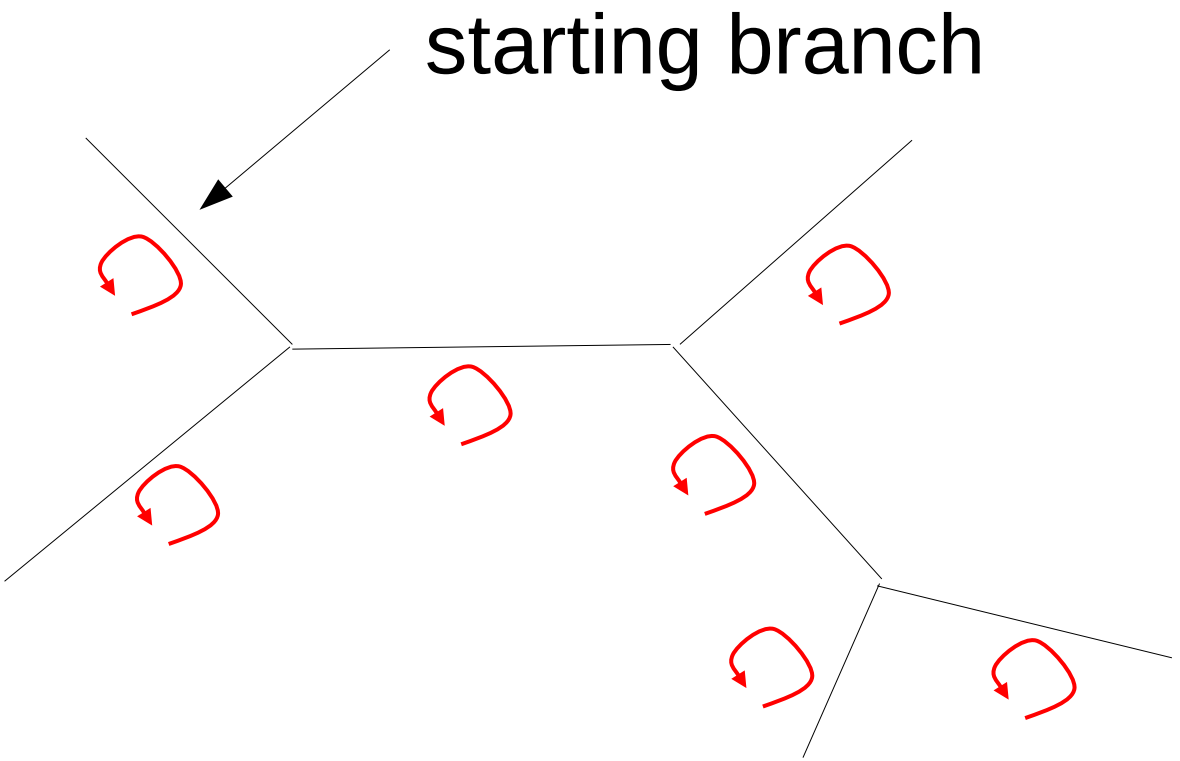
Branch Length Optimization



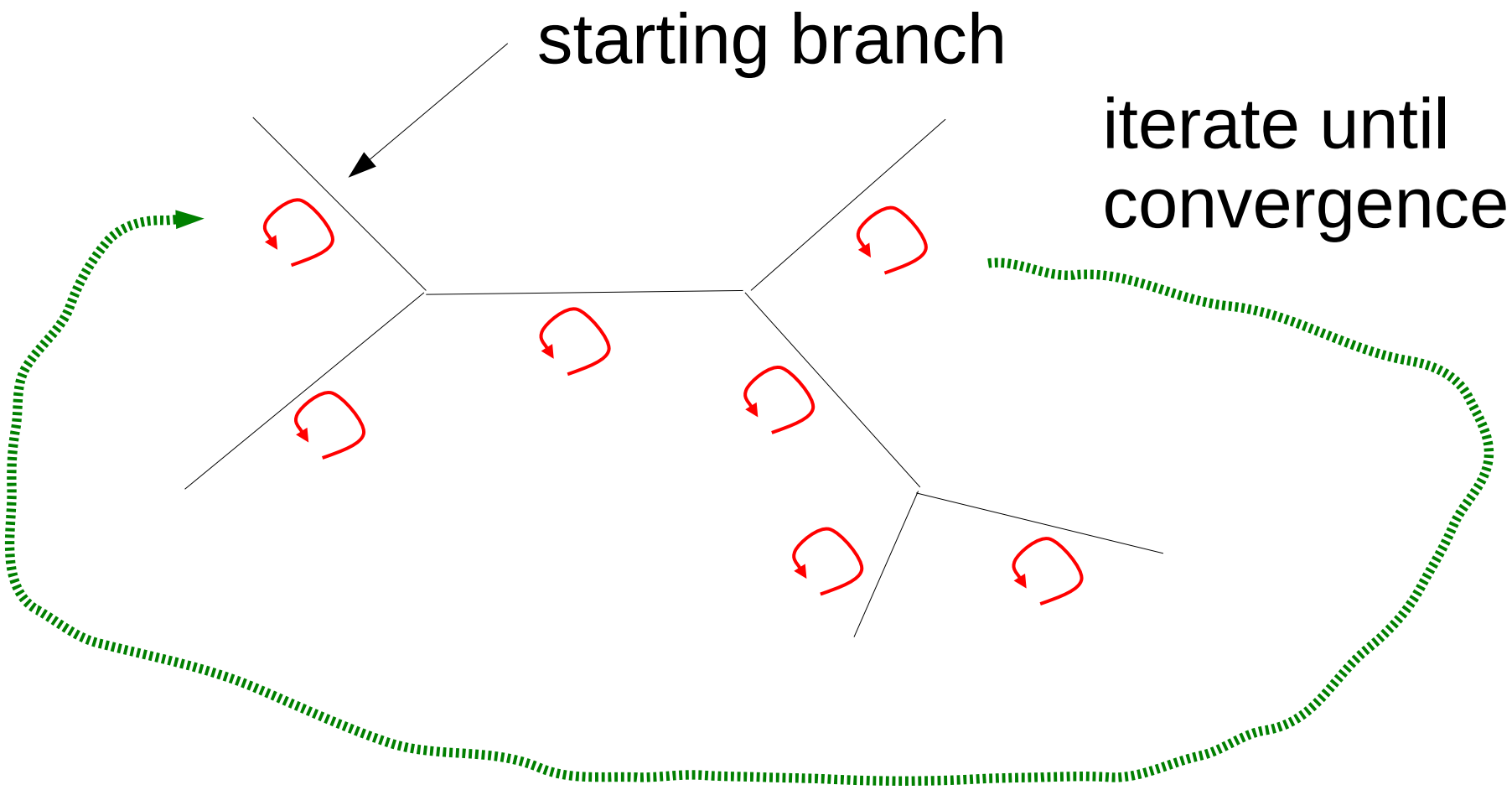
Branch Length Optimization



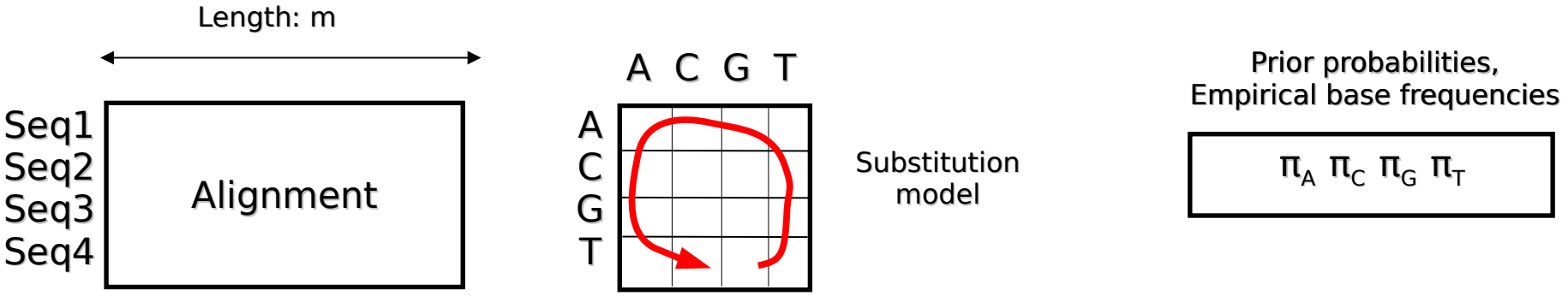
Branch Length Optimization



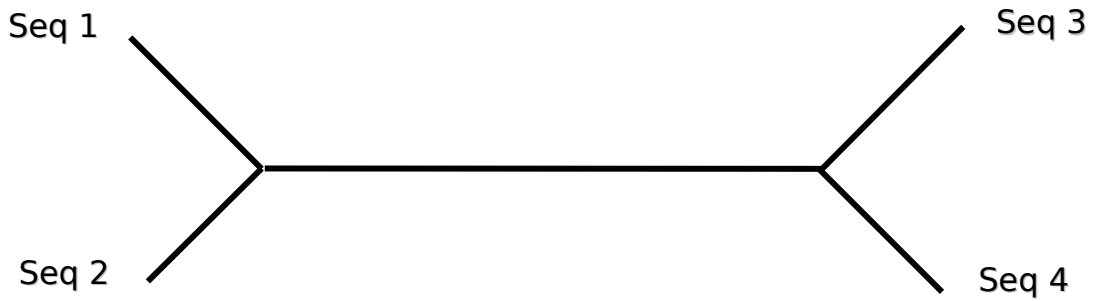
Branch Length Optimization



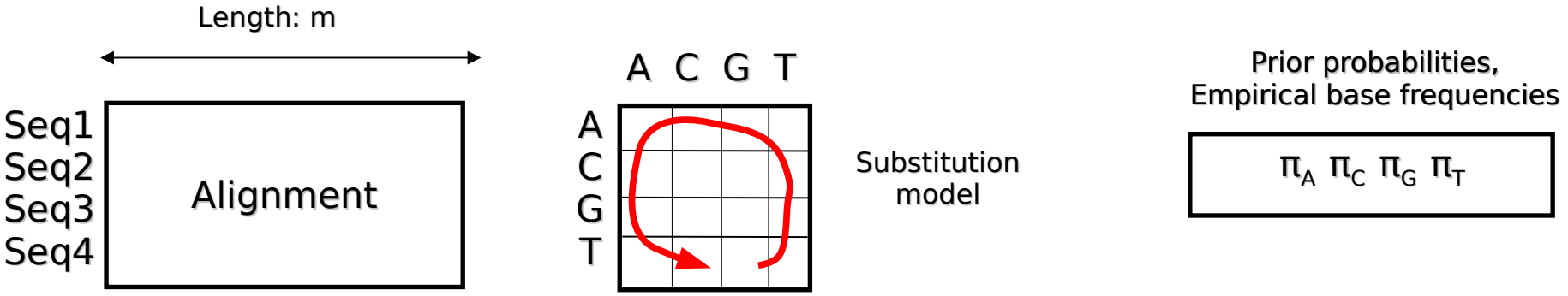
Maximum Likelihood



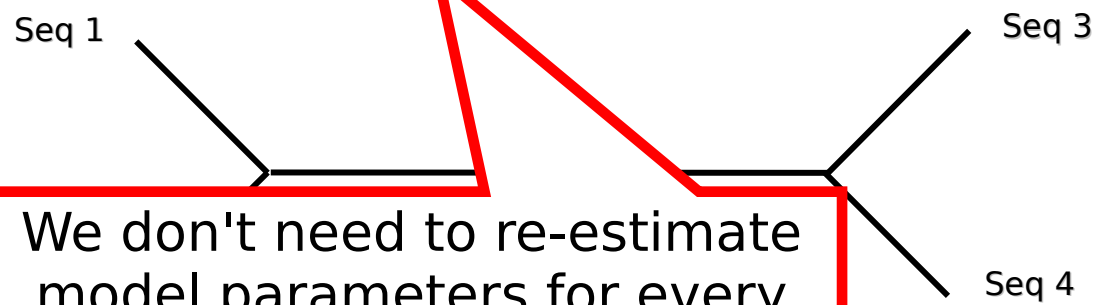
optimize model parameters



Maximum Likelihood

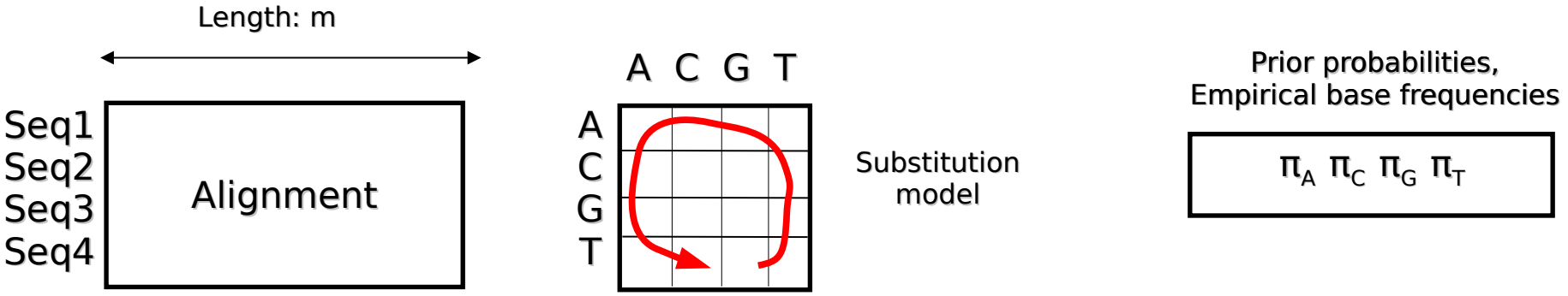


optimize model parameters

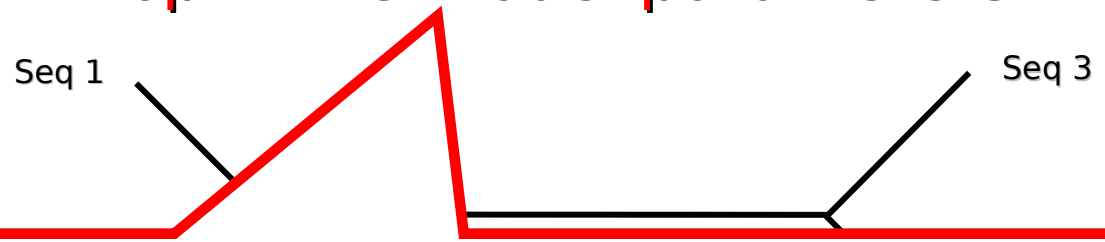


We don't need to re-estimate model parameters for every tree topology as long as the tree is reasonable, i.e., non-random

Maximum Likelihood



optimize model parameters



Methods used for model parameter optimization (other than branch lengths)

1. BFGS
2. Brent's method
3. Expectation maximization approaches

Numerical Optimization Procedures

- See chapters 9 & 10 of: *Numerical Recipes in C – The Art of Scientific Computing*

Basic Operations
Maximum Likelihood

- Compute Conditional Likelihood Vector at an inner node
- Compute Likelihood at Virtual Root
- Optimize a Branch Length for a given Branch
- Optimize all Branch Lengths
- Optimize other Model Parameters

Basic Operations
Maximum Likelihood

- Compute Conditional Likelihood Vector at an inner node
- Compute Likelihood at Virtual Root
- Optimize a Branch Length for a given Branch
- Optimize all Branch Lengths
- Optimize other Model Parameters

The optimizers are the tricky routines!

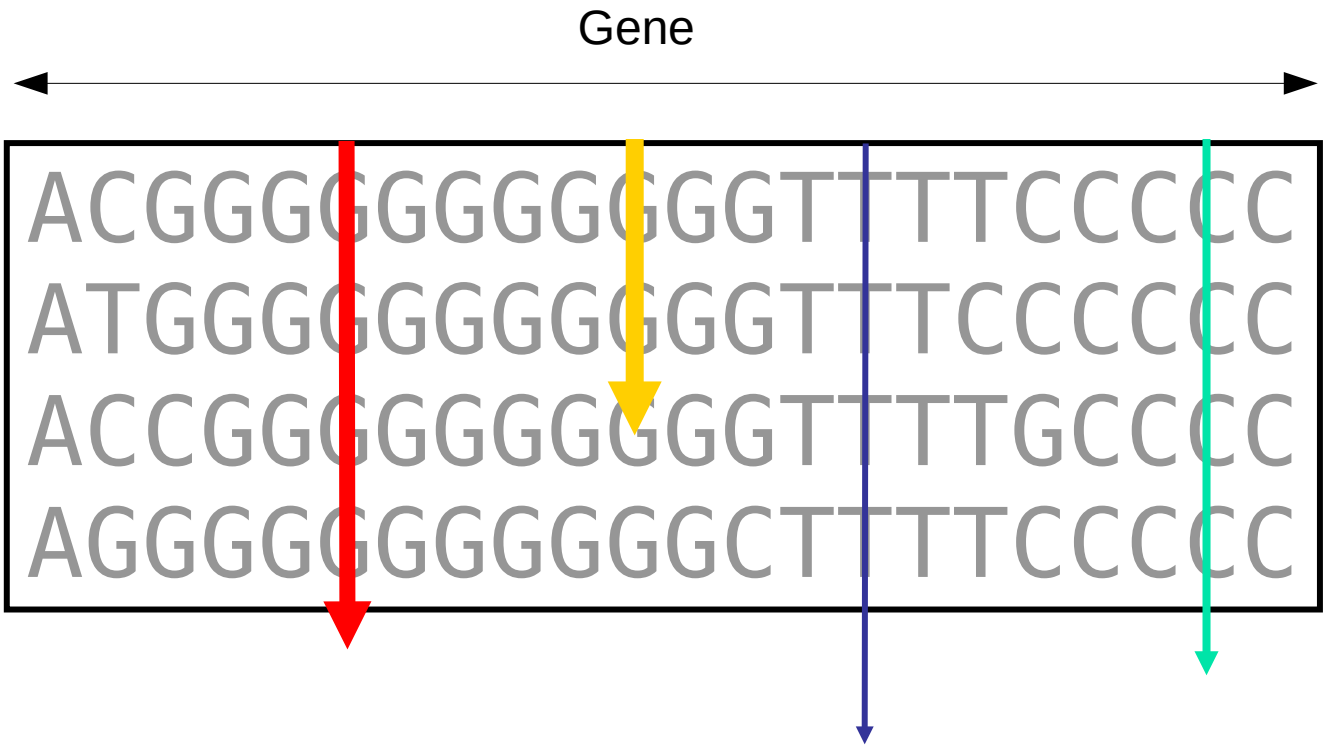
Outline

- Last time:
  - How to Compute the Likelihood of a tree
  - How to compute the Likelihood efficiently: Felsenstein Pruning Algorithm
- Today & next time
  - What is hidden in P(t) – what do the models look like ?
  - How to compute the Maximum Likelihood score on a tree?
  - **Advanced substitution models**
  - Efficiently computing the Likelihood on trees
  - Parallel Likelihood computations

Protein Substitution Models

- The GTR *Q* matrix for protein data has 189 free parameters instead of just 5 (DNA)
- Estimating **189** rate parameters is difficult, time-consuming, and may lead to over-parameterizing the model
- Instead, empirical models such as `JTT, LG, WAG, MTMAM`, etc. are used
- The *Q* matrices are obtained by jointly optimizing model parameters on a large collection of reference alignments
- The models differ with respect to:
  - the amount of data used to obtain them
  - the type of data on which the models have been optimized
    - e.g., dedicated models for HIV, FLU, Mammals
  - the numerical optimization methods used
- Examples of general models:
  - **`LG`**: Le & Gascuel: “An Improved General Amino Acid Replacement Matrix”
  - **`WAG`**: Whelan & Goldman: “A General Empirical Model of Protein Evolution Derived from Multiple Protein Families Using a Maximum-Likelihood Approach”

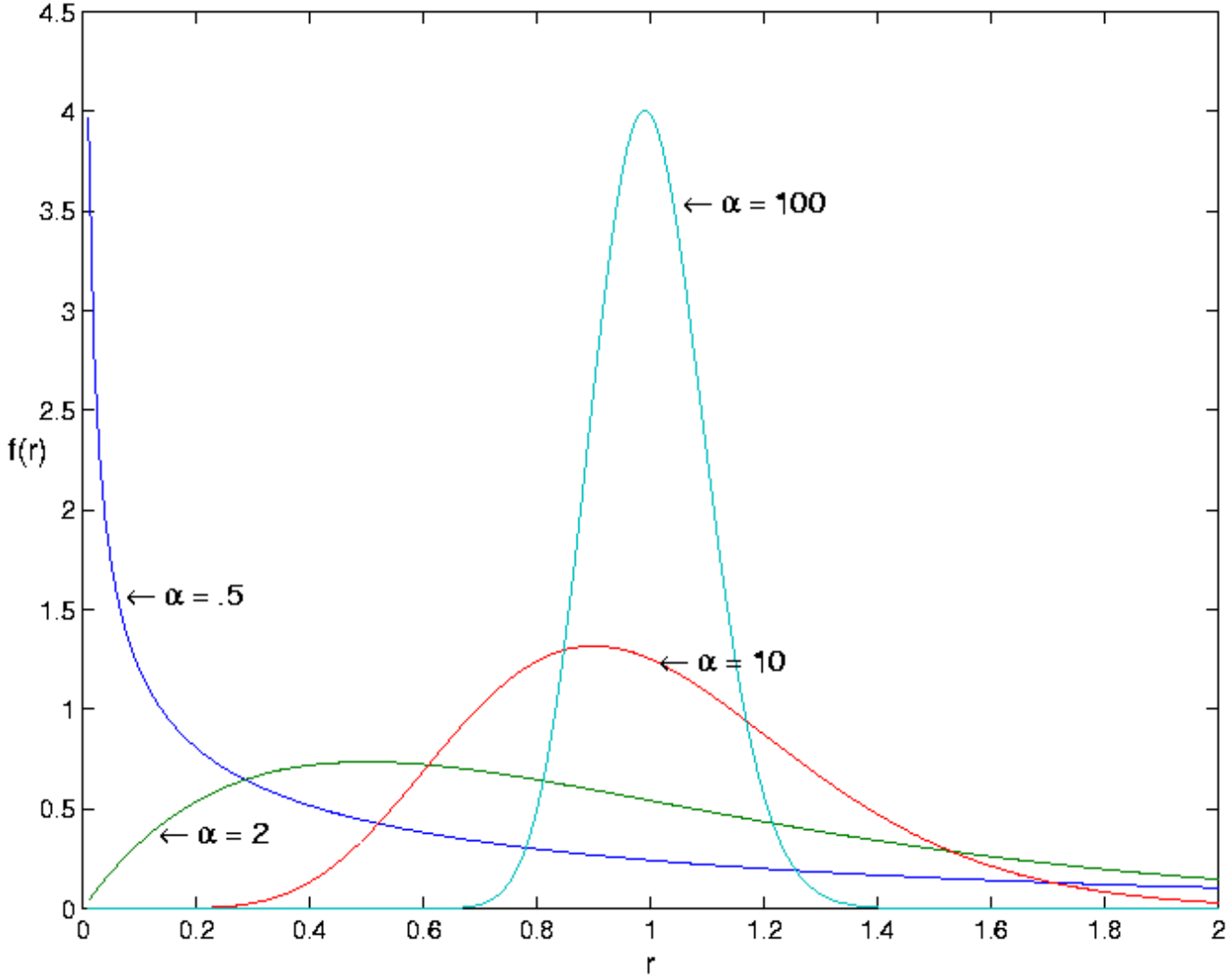
Rate Heterogeneity among Sites



- Among-site rate heterogeneity
  - Biological phenomenon

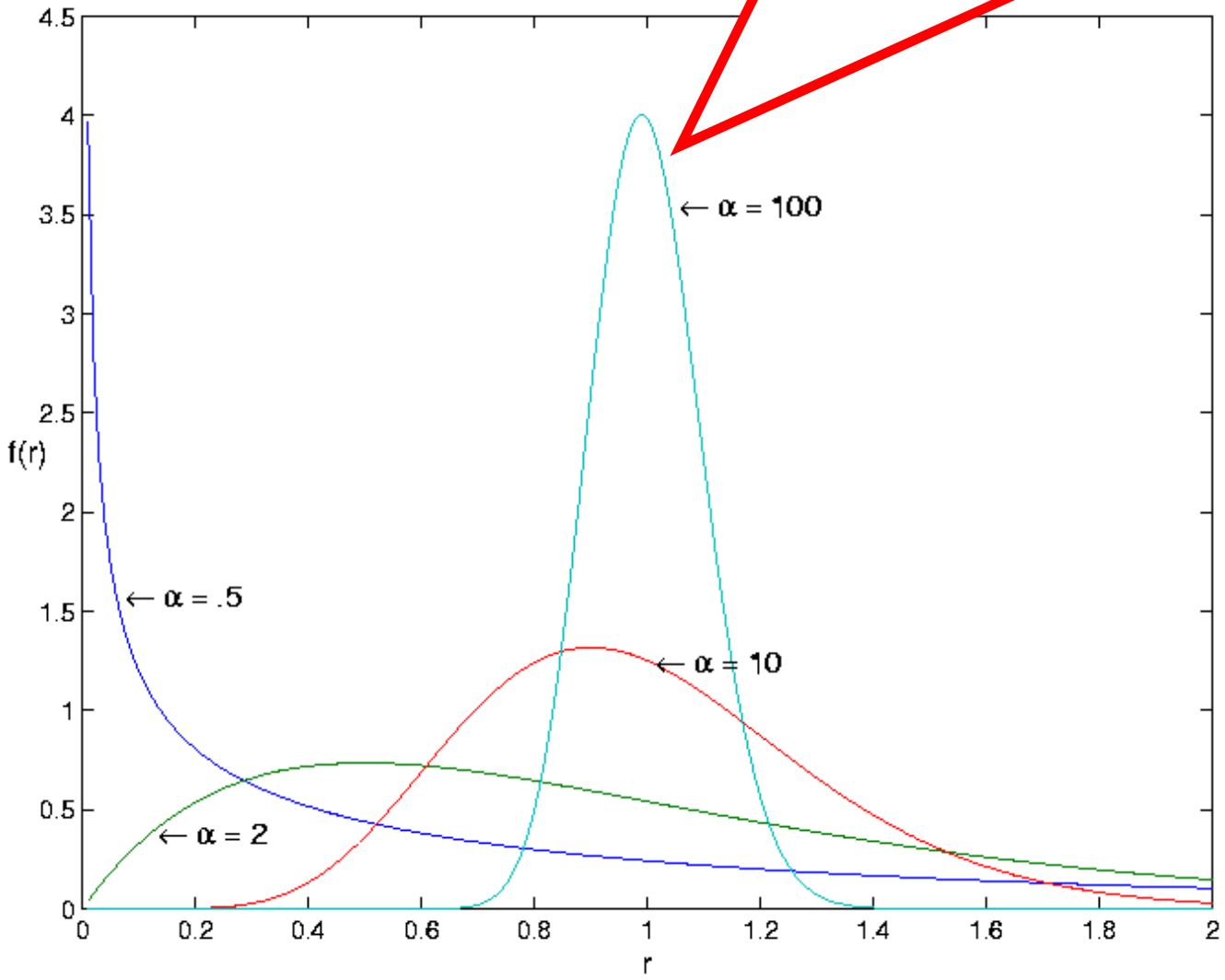
    → different sites/columns evolve at different speeds
  - Need to accommodate this in our models

Γ-Distribution

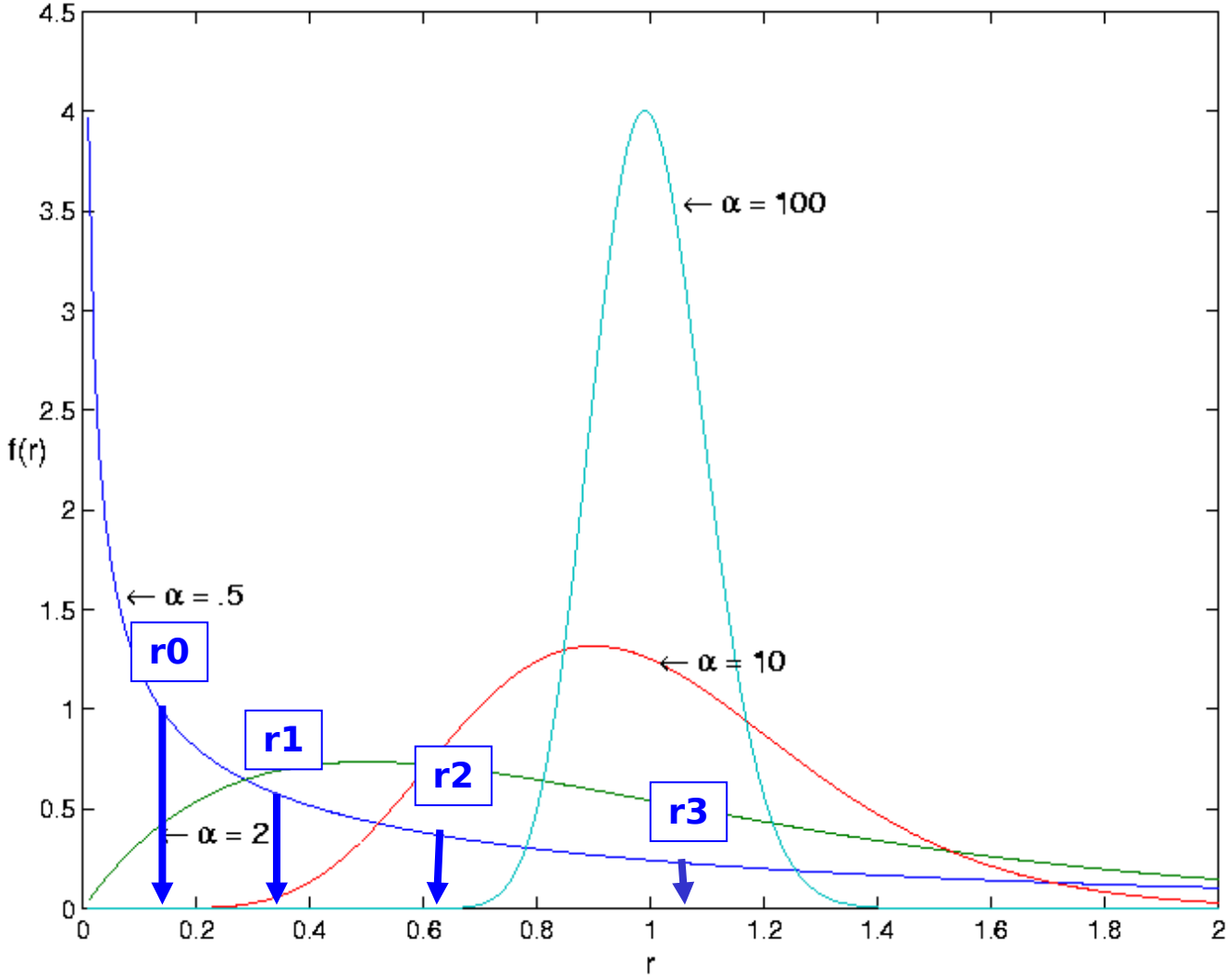


Γ-Distribution

Small α ➔high rate heterogeneity
Large α ➔low rate heterogeneity



Discrete Γ-Distribution



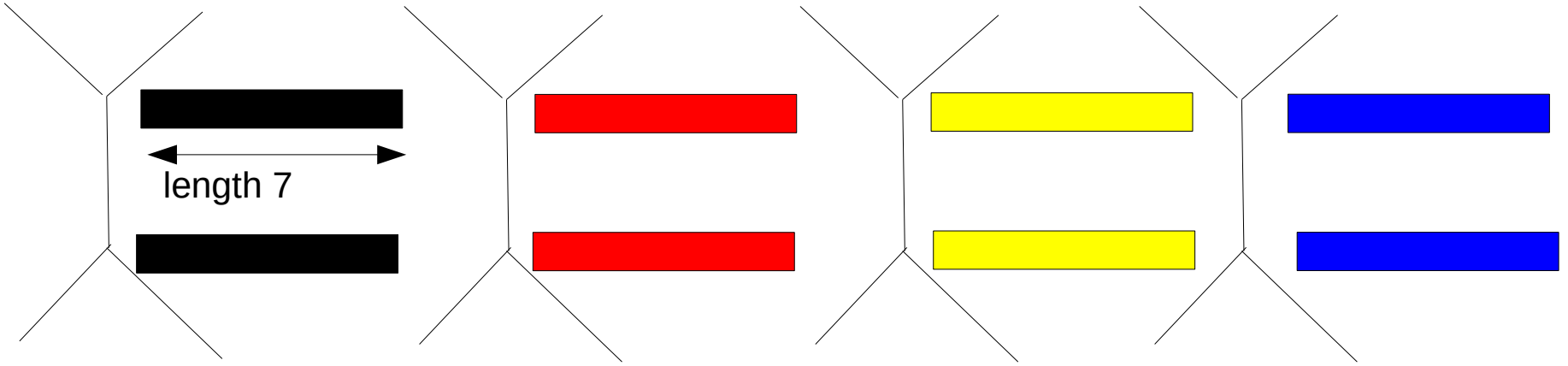
An Abstract View of Γ

rate 0
$P(t) = e^{Qr_0 t}$

rate 1
$P(t) = e^{Qr_1 t}$

rate 2
$P(t) = e^{Qr_2 t}$

rate 3
$P(t) = e^{Qr_3 t}$



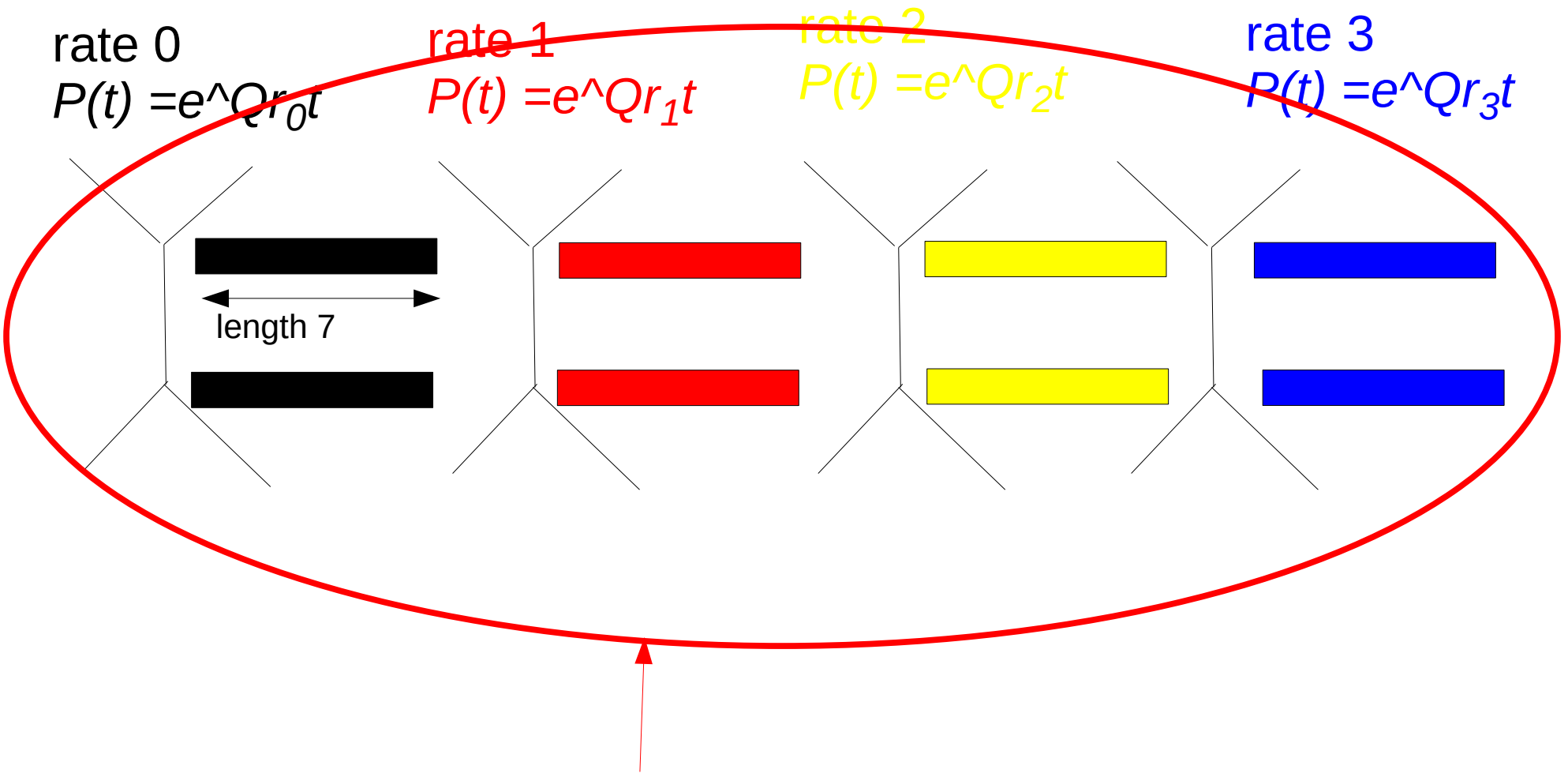
This is the integral of the likelihood we approximate via discretization

$$LnL(i) = \log(\tfrac{1}{4} * (L_0 + L_1 + L_2 + L_3))$$

Log likelihood at site *i*

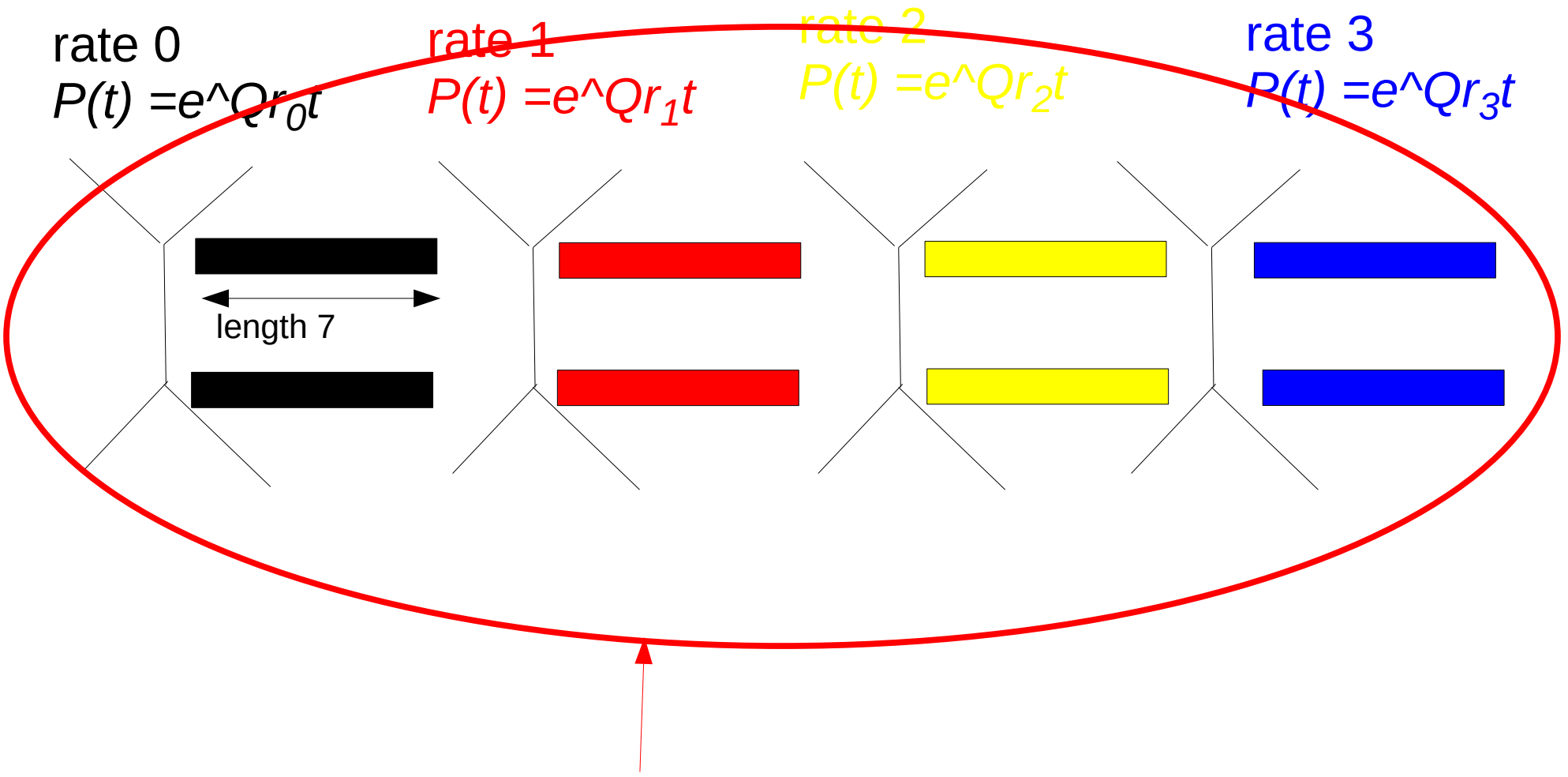
All Γ rates have equal probability

An Abstract View of Γ



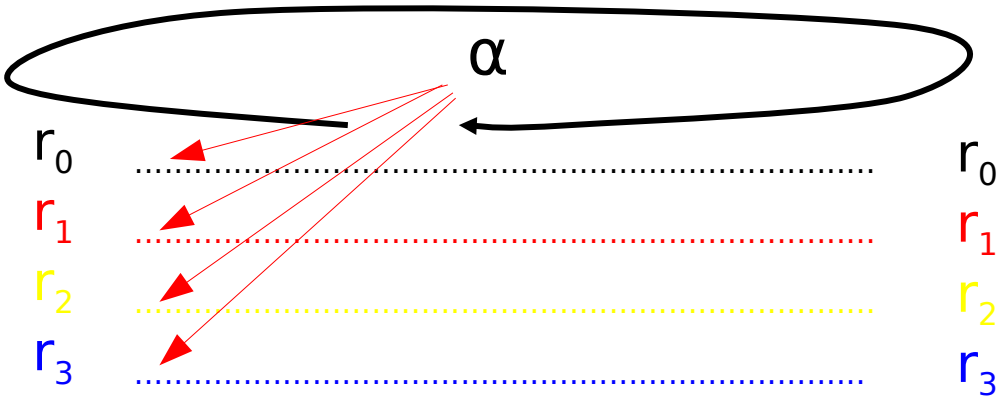
4 times higher memory consumption

An Abstract View of Γ



4 times more floating point operations

Γ Model of Rate Heterogeneity with 4 discrete rates




```
Species209   NNNNNNNNNNN NNNNNNNNNNN NNNNNNNNNNN NNNNNNNNNNN
Species159   UAUCUCGUUG AUCCUGCCAG UAGUAUNUGC UUGUCUCAAA
Species109   NNNNNNNNNNN NNNNNNNNNNN NNNNAUAUGC UUGUCUC--A
Species025   NNNNNNNNNNN NNNNNNNNNNN NNNNNNNNNNN NNNNNNNAAA
Species004   NNNNNNNNNNN NNNNNNNNNNN NNNNNNNNNNN NNNNNNNNNN
Species186   UACCUCGUUG AUCCUGCCAG UAGUAUAUGC UUGUCUCAAA
Species034   NNNNNNNNNNN NNNNNNNNNNN NNNNNNNNNNN NNNNNNNAAA
Species008   NNNNNNNNNNN NNNNNNNNNNN NNNNNNNNNNN NNNNNNNAAA
Species192   NNNNNNNNNNN NNNNNNNNNNN NNNNNNNNNNN NNNNNNCAAA
Species132   NNNNNNNNNNN NNNNNNNNNNN UAGUAUAUGC UUGUCUCAAA
Species172   NNNNNNNNNNN NNNGAAUU-C UAUUANANGC -UGUUUCAAA
```

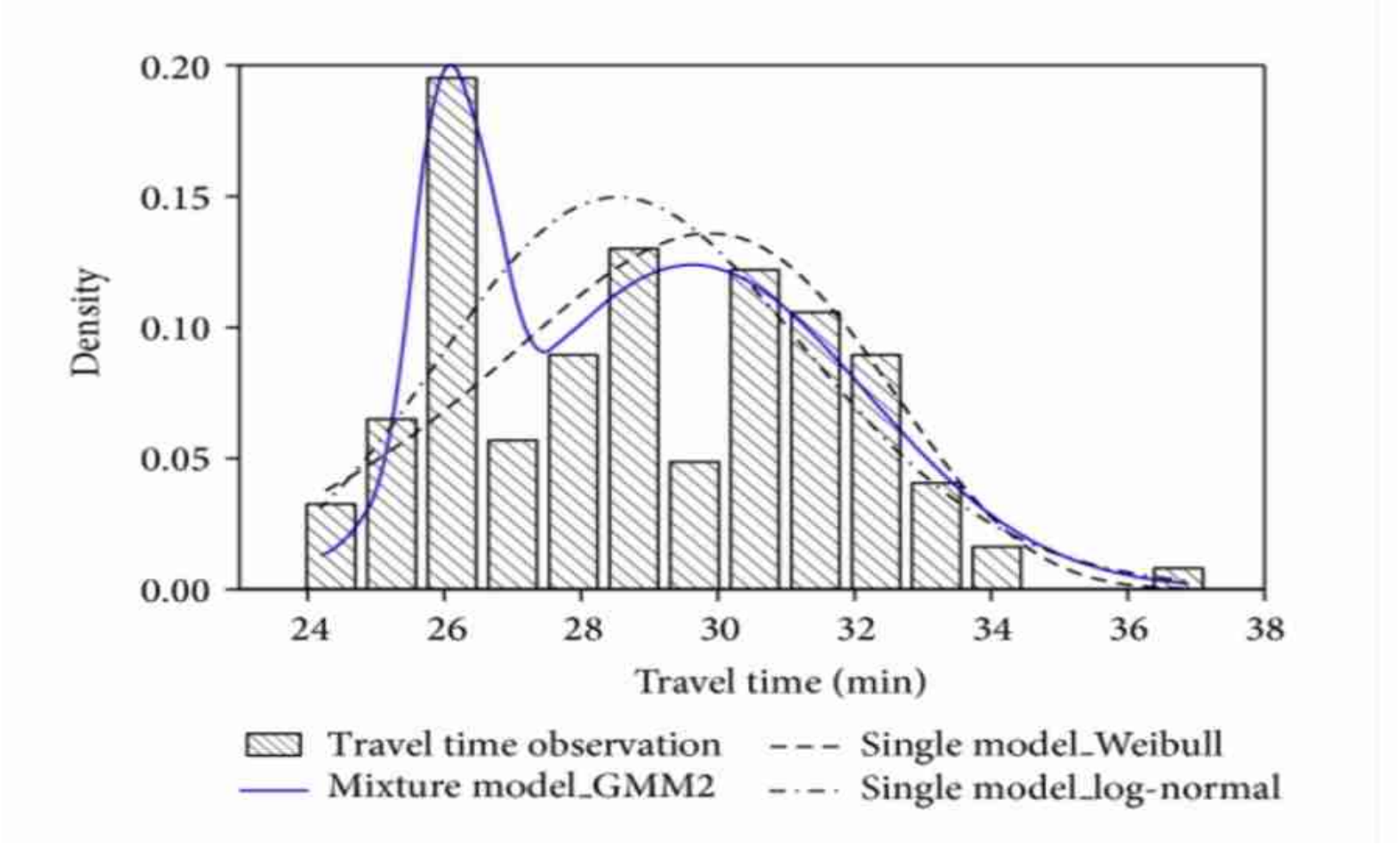
Mixture Models

- The Γ model of rate heterogeneity is a simple example of so-called mixture models
- From Wikipedia: “In statistics, a mixture model is a probabilistic model for representing the presence of subpopulations within an overall population, without requiring that an observed data set should identify the sub-population to which an individual observation belongs. Formally a mixture model corresponds to the mixture distribution that represents the probability distribution of observations in the overall population.”
- The Γ model gives us 4 discrete evolutionary rates over which we integrate (add) the likelihood for each site, without assigning a specific rate to a specific site

Mixture Models

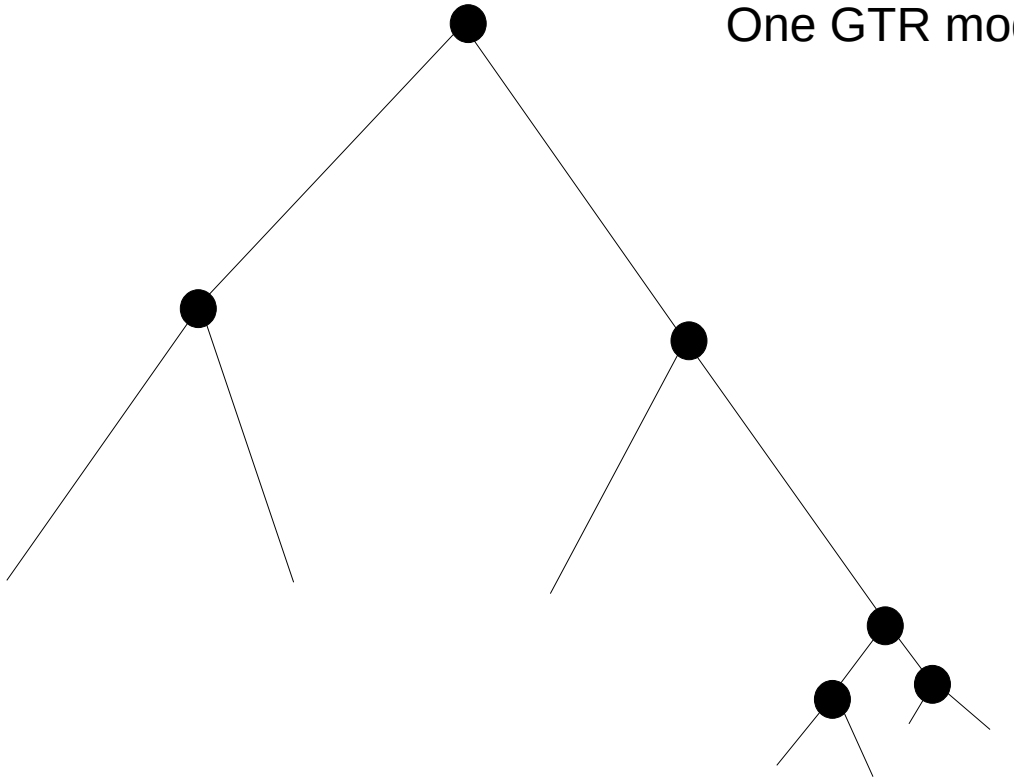
- We can also imagine to integrate the likelihood over a set of
  - distinct *Q* matrices
  - distinct base frequencies
  - or combinations thereof
- The LG protein substitution model is an example thereof:
- It uses 4 distinct empirical *Q* matrices and 4 distinct sets of base frequencies π over which we integrate just like for the Γ model

An example



Taken from: “Measuring Service Reliability Using Automatic Vehicle Location Data”
→ bus service reliability

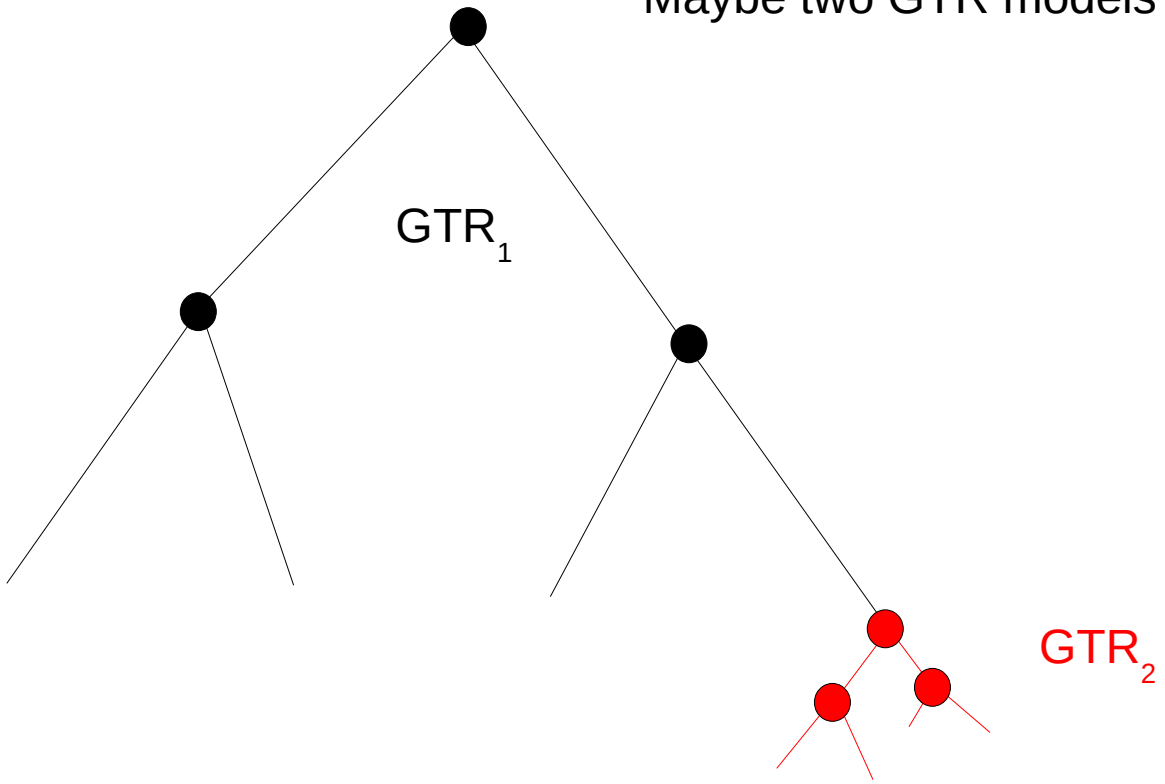
Heterotachous Models



One GTR model for the entire tree

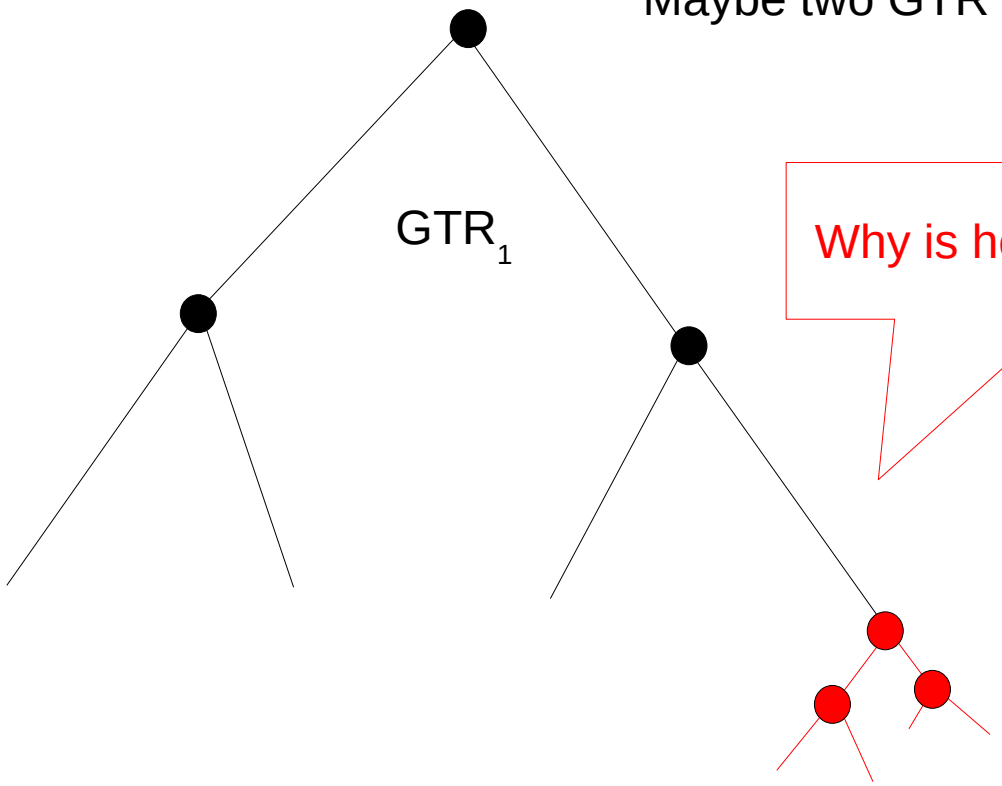
Heterotachous Models

Maybe two GTR models describe this better?



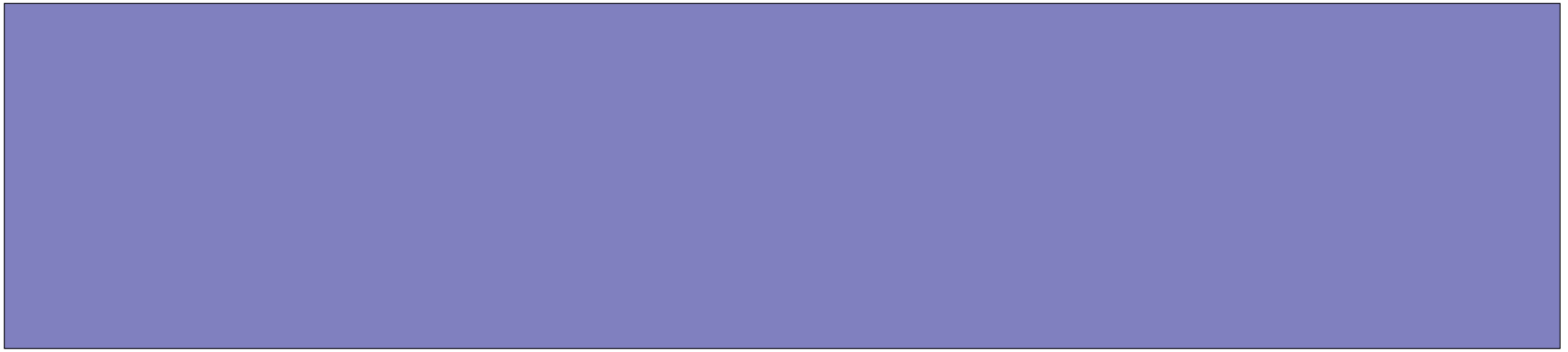
Heterotachous Models

Maybe two GTR models describe this better?



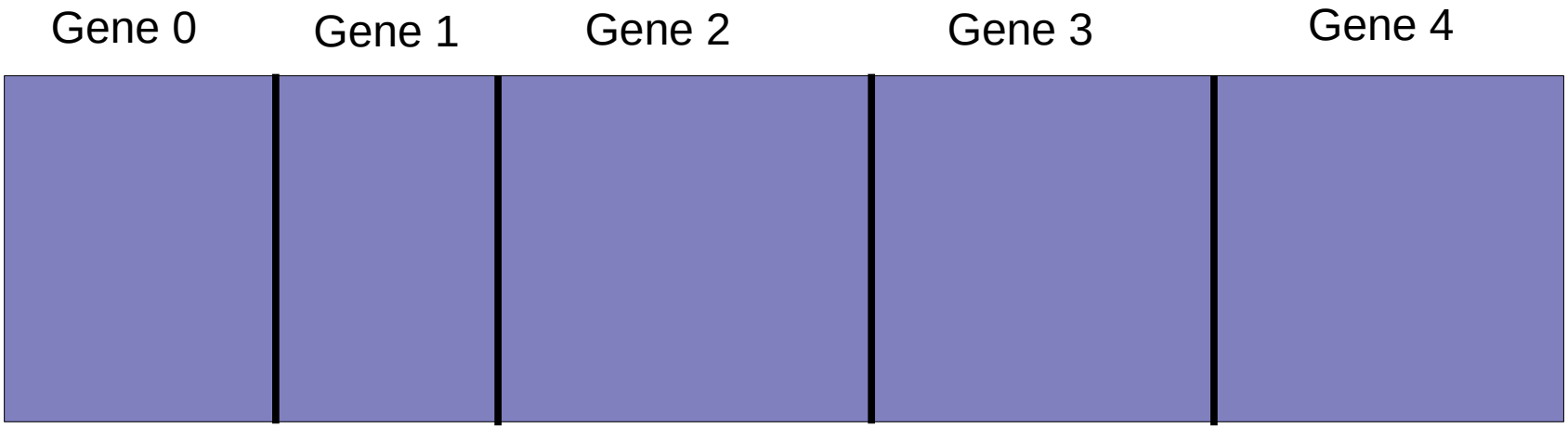
Why is heterotachy difficult?

What is a partitioned dataset?



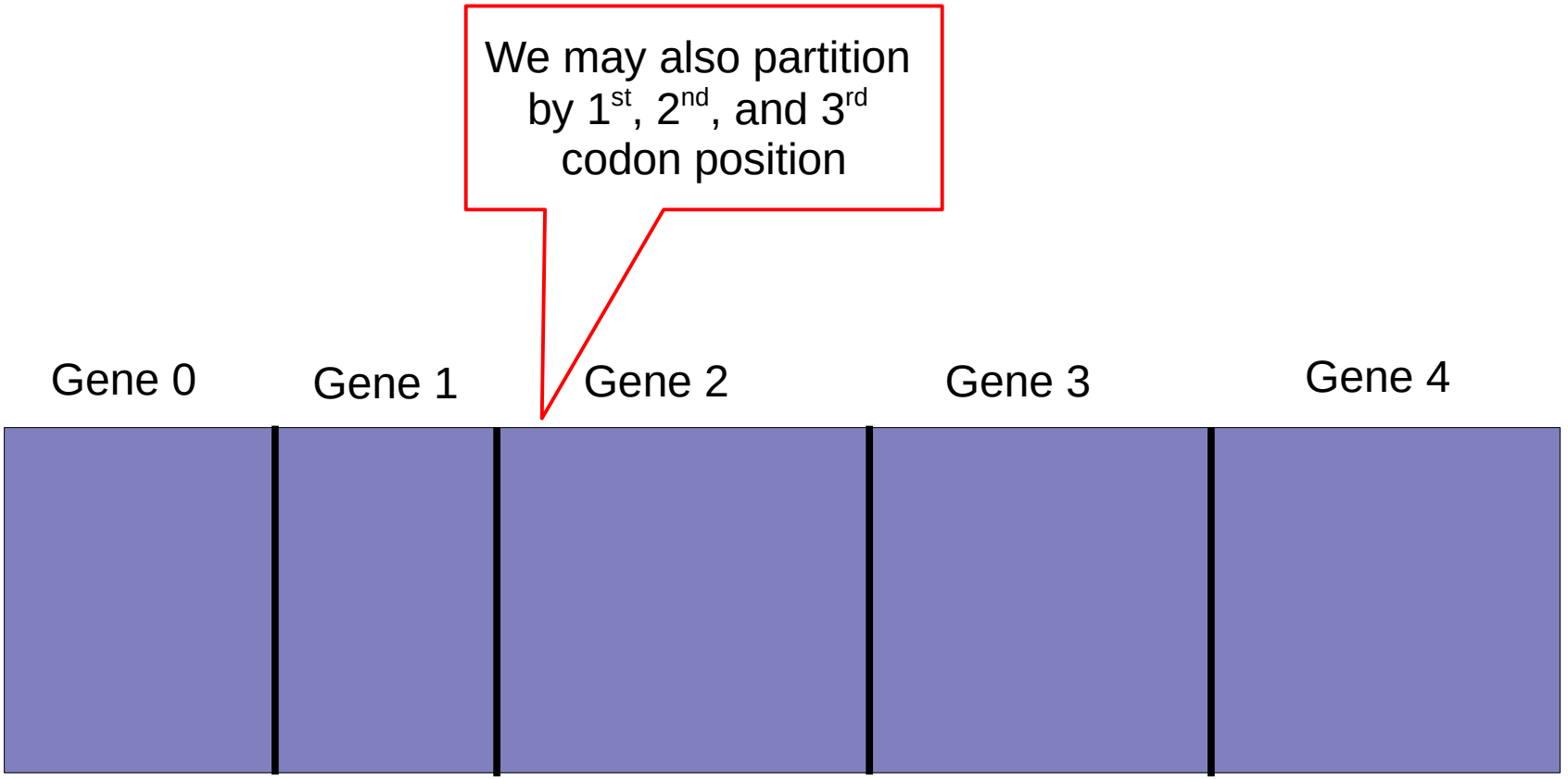
Multi-gene or whole-genome alignment

What is a partitioned dataset?



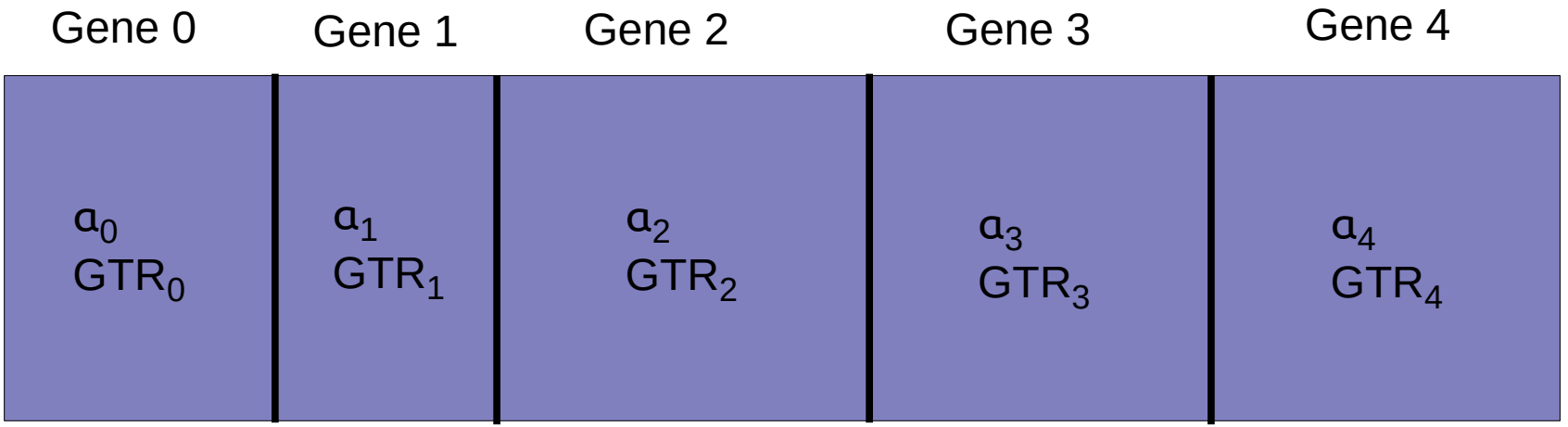
Multi-gene or whole-genome alignment

What is a partitioned dataset?

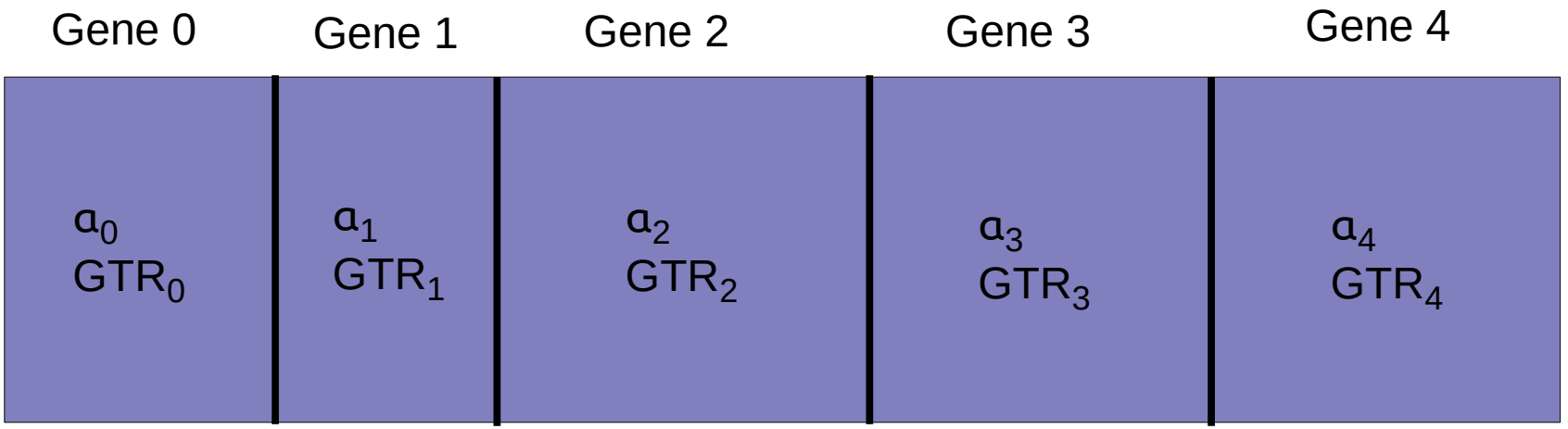


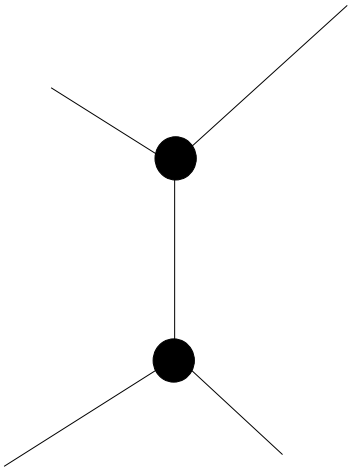
Multi-gene or whole-genome alignment

What is a partitioned dataset?



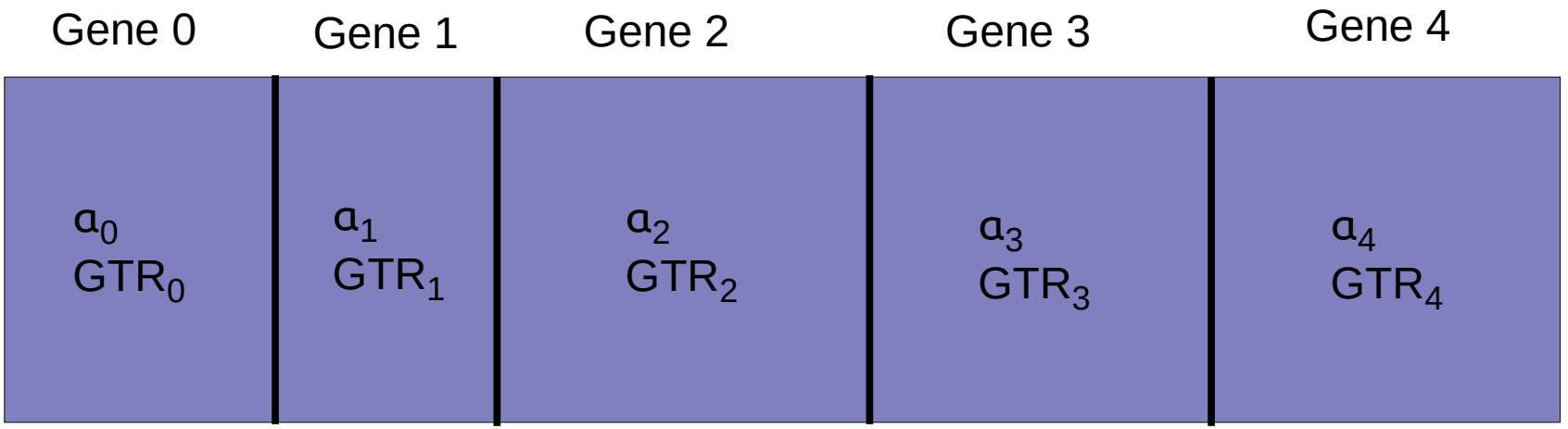
What is a partitioned dataset?

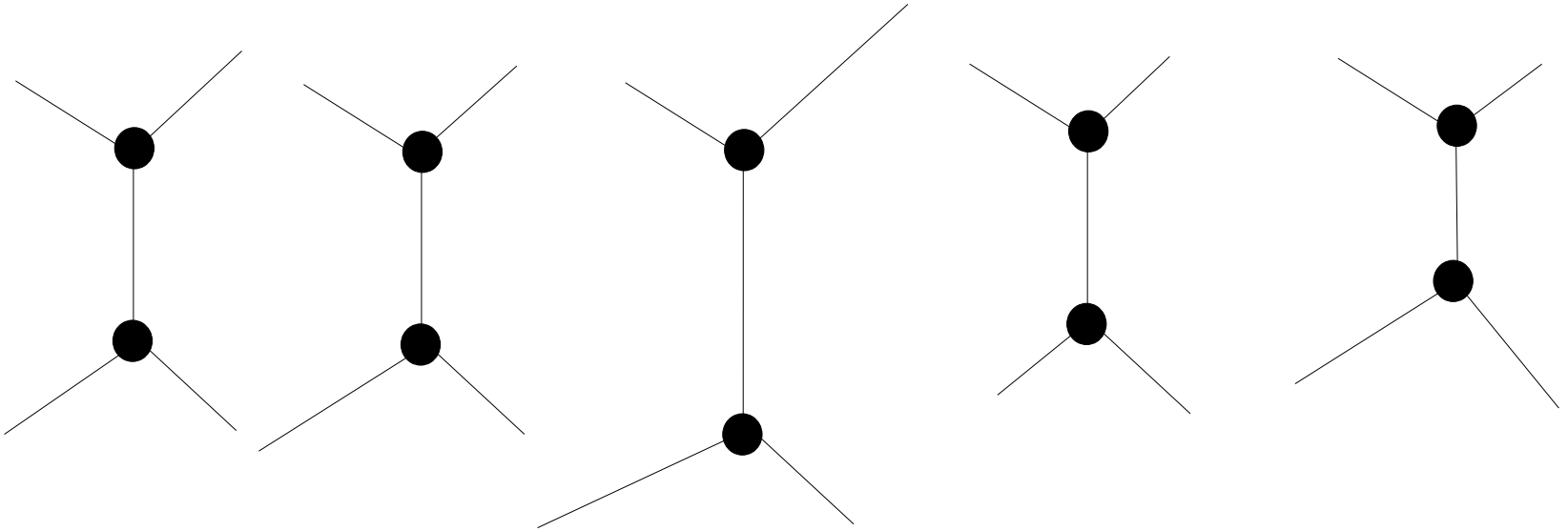




Joint branch length estimate

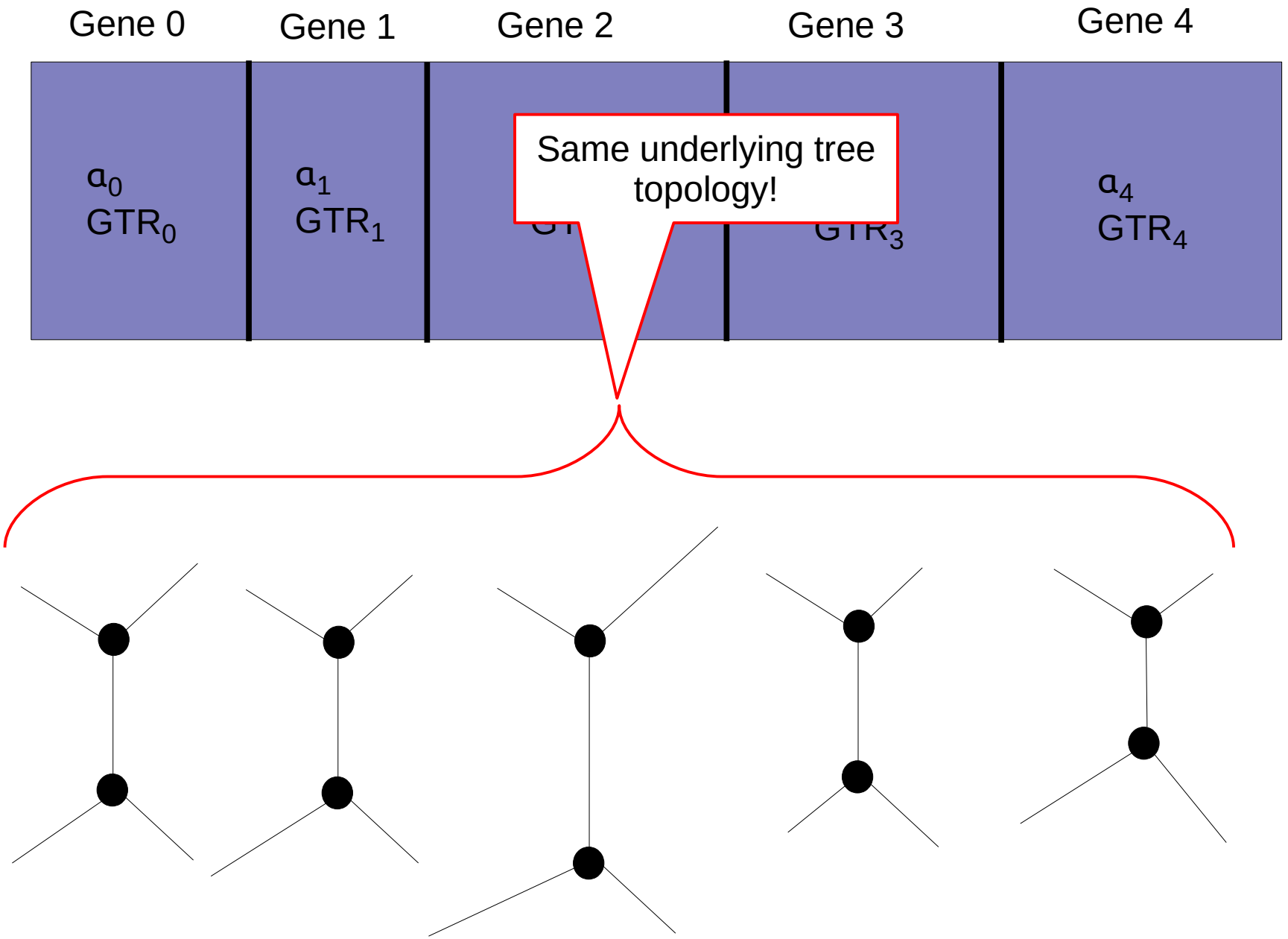
What is a partitioned dataset?



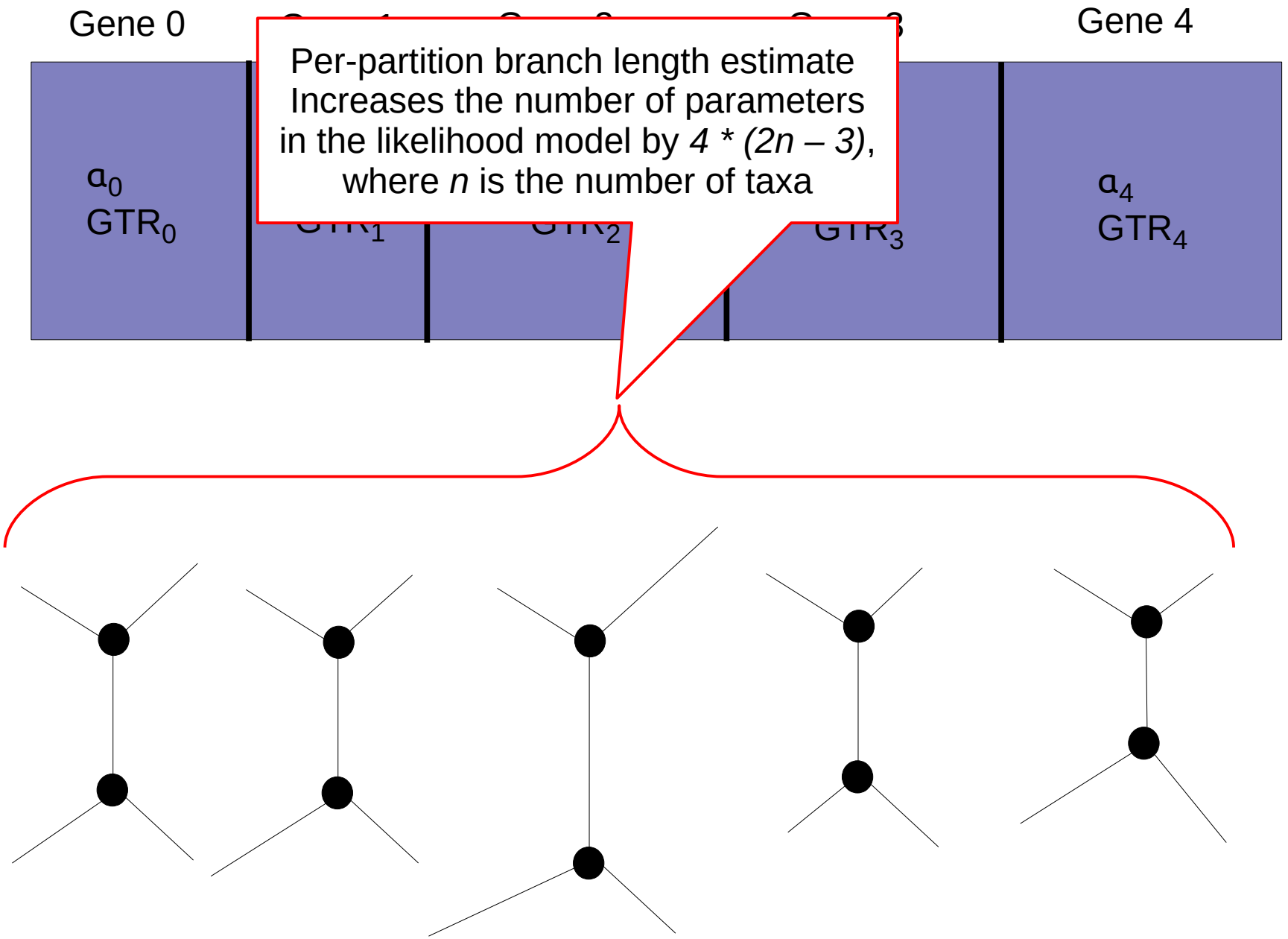


per-partition branch length estimate

What is a partitioned dataset?



What is a partitioned dataset?



Models and Parameters

- If we add an additional parameter to a model, the likelihood will become better
- However, this does not mean anything, as
  - We might be over-parameterizing
  - The key question is if the more complex model yields a different tree topology
- So, how do we determine the best-fit model for a given dataset?

Nested models

- A particular model is said to be nested within a more complex model only if constraining parameter values of the latter yields the former!
- So, the model can only be constrained in one direction to determine if its nested!
- If I need to constrain both models for which I intend to assess nesting, they are not nested.
- Example: The *F81* (equal rates, unequal stationary frequencies) and *K2P* (2 distinct rates, equal stationary frequencies) models are not nested within each other.

  → This is because fixing the parameter values of either model does not yield the other model
- However, they are both nested within *GTR*

Model Testing

- If models are nested we can use a likelihood ratio test
- Model *A* is nested in model *B* if parameters in model *A* are a subset of the parameters in model *B*
- For instance: the *Jukes Cantor (JC)* model is nested in the *General Time Reversible (GTR)* model of nucleotide substitution
- $LR = P(\mathbf{D}|A) / P(\mathbf{D}|B) = L(A) / L(B)$
- $\Delta = ln(LR^2) = 2\,(ln(L(A)) - ln(L(B))$
- Compare Δ to x^2 distribution with $k_A - k_B$ degrees of freedom to determine if the Δ is significant or not
- The degrees of freedom difference is the difference in the number of free parameters in the models
- How many free parameters do the *JC* and *GTR* models have?

Model Testing

- If models are nested we can use a likelihood ratio test
- Model *A* is nested in model *B* if parameters in model *A* are a subset of the parameters in model *B*
- For instance: th… *me Reversible (GTR)* model of…
- *LR = P(**D**|A) / P(**D**|B) =* …
- *Δ = ln(LR²) = 2 (ln(L(A)) – ln(L(B))*
- Compare *Δ* to x^2 distribution with $k_A - k_B$ degrees of freedom to determine if the *Δ* is significant or not
- The degrees of freedom difference is the difference in the number of free parameters in the models
- How many free parameters do the *JC* and *GTR* models have?

  → JC: 0

  → GTR: 8

We are only allowed to compare likelihoods on the same data **D**!

What if Models are not nested?

- One can use other criteria such as
  - *Akaike Information Criterion (AIC)*
  - *Bayesian Information Criterion (BIC)*
- I will spare you the details, but the basic idea always is:
  - Compute likelihood of alternative models
  - Penalize the more parameter-rich models

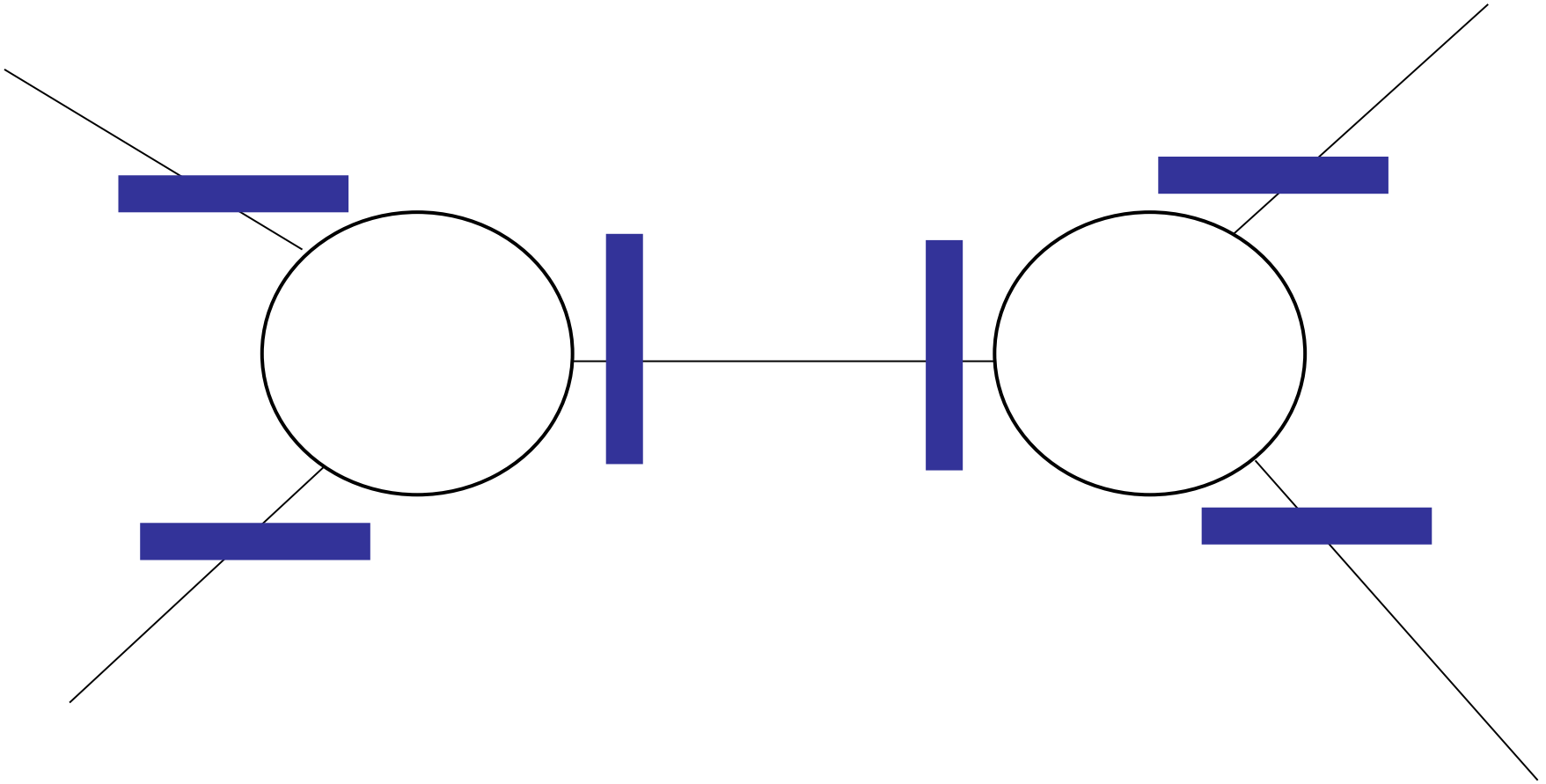
Outline

- Last time:
  - How to Compute the Likelihood of a tree
  - How to compute the Likelihood efficiently: Felsenstein Pruning Algorithm
- Today & next time
  - What is hidden in P(t) – what do the models look like ?
  - How to compute the Maximum Likelihood score on a tree?
  - Advanced substitution models
  - **Efficiently computing the Likelihood on trees**
  - Parallel Likelihood computations

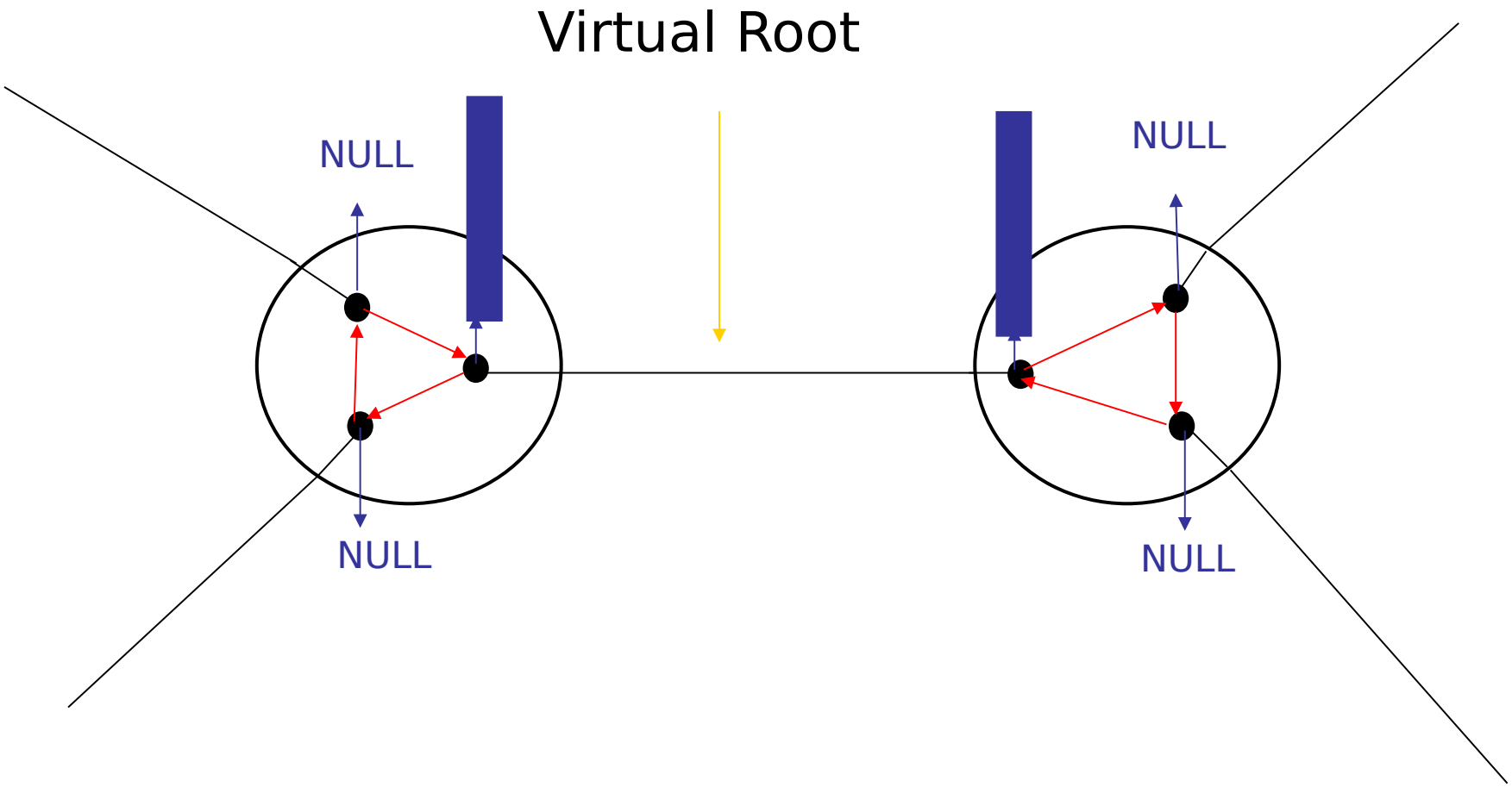
Data Structures for unrooted Trees

- Unrooted trees with dynamically changing virtual roots need a dedicated tree data structure
- Why can the virtual root positions change dynamically?
- If we apply a topological move (NNI, SPR, TBR) will we have to re-compute all conditional likelihood vectors?

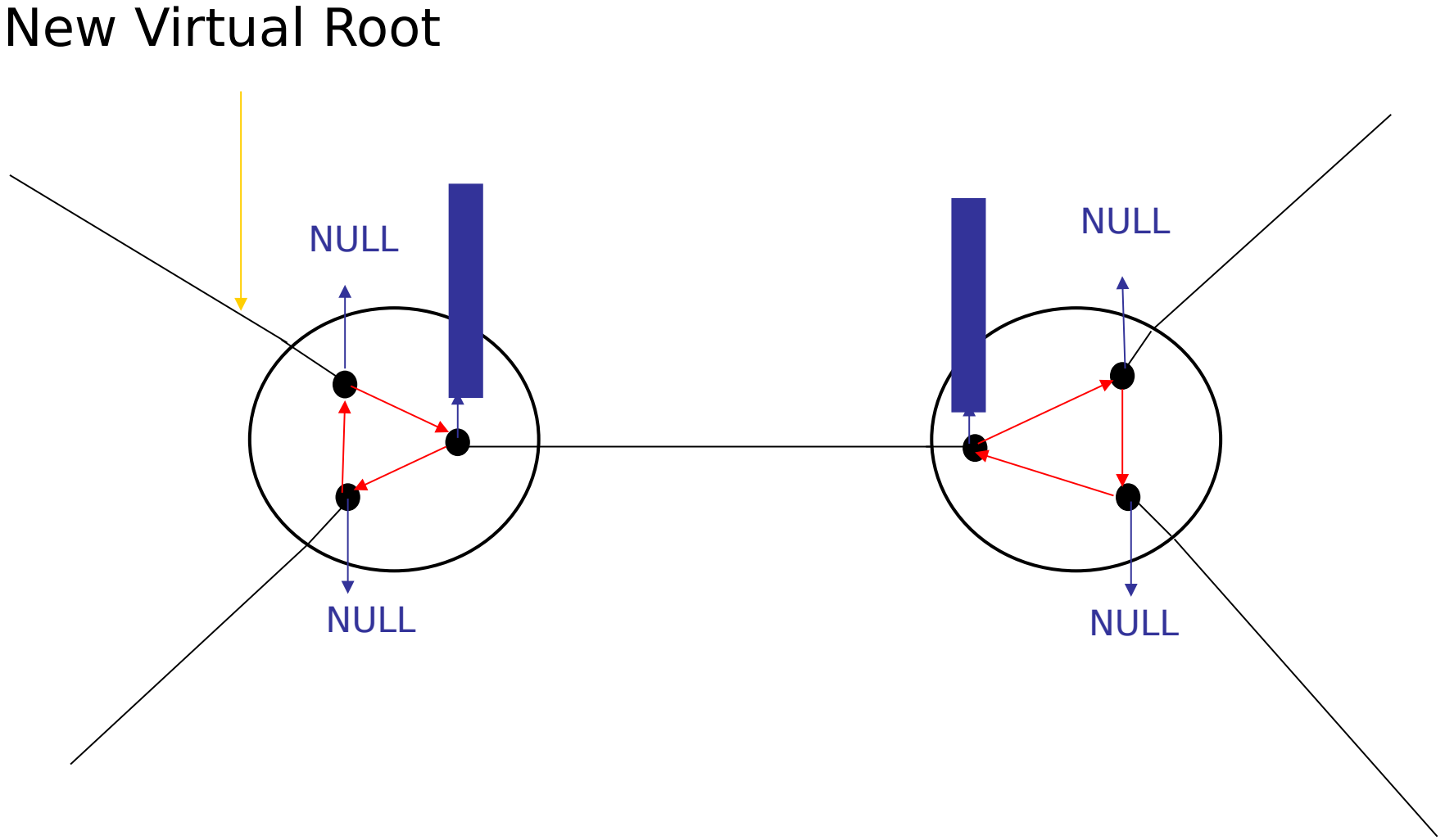
Memory Organization: Conditional Likelihood Vectors with an Unrooted View



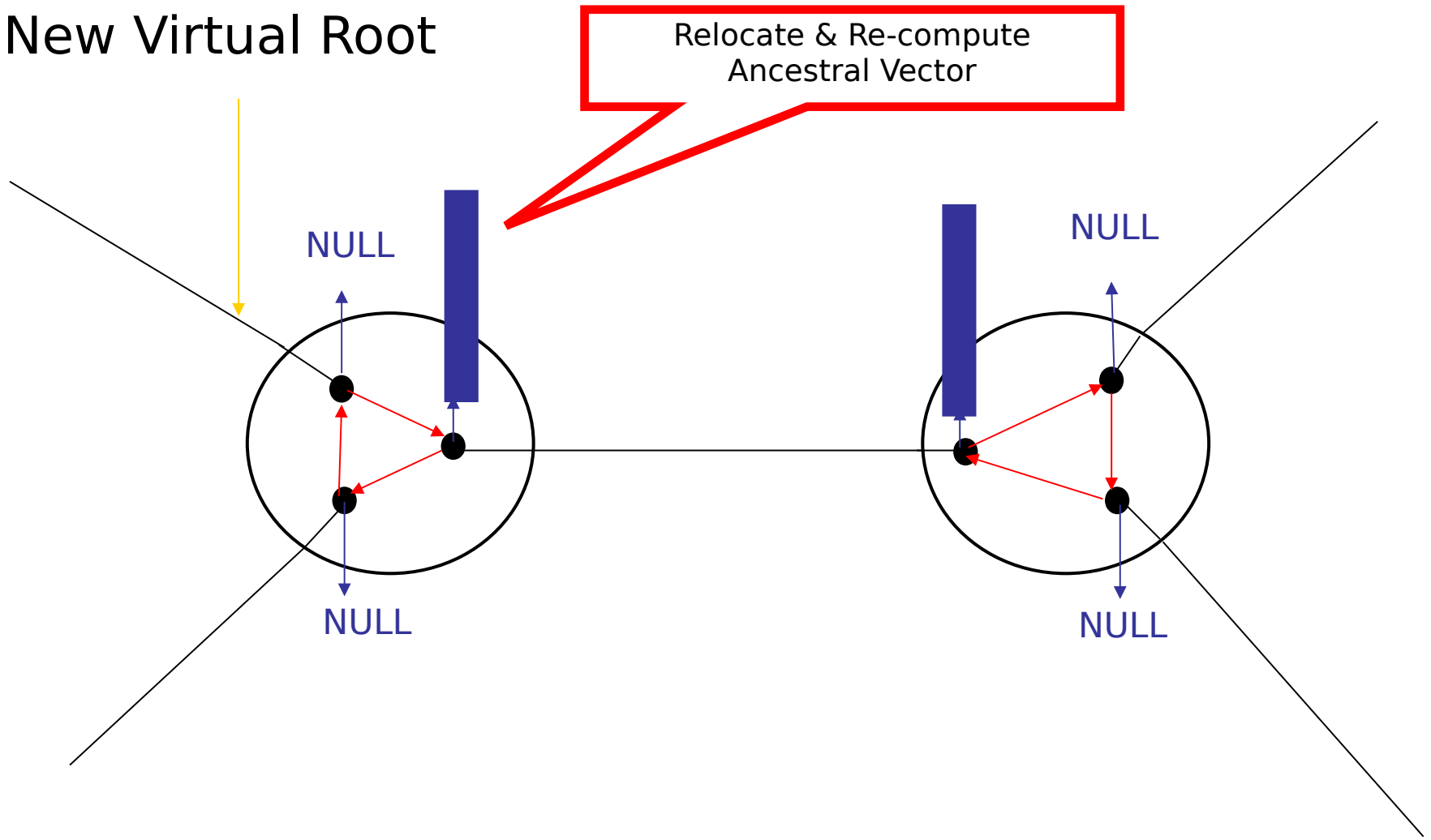
Memory Organization: Conditional Likelihood Vectors with a Rooted View



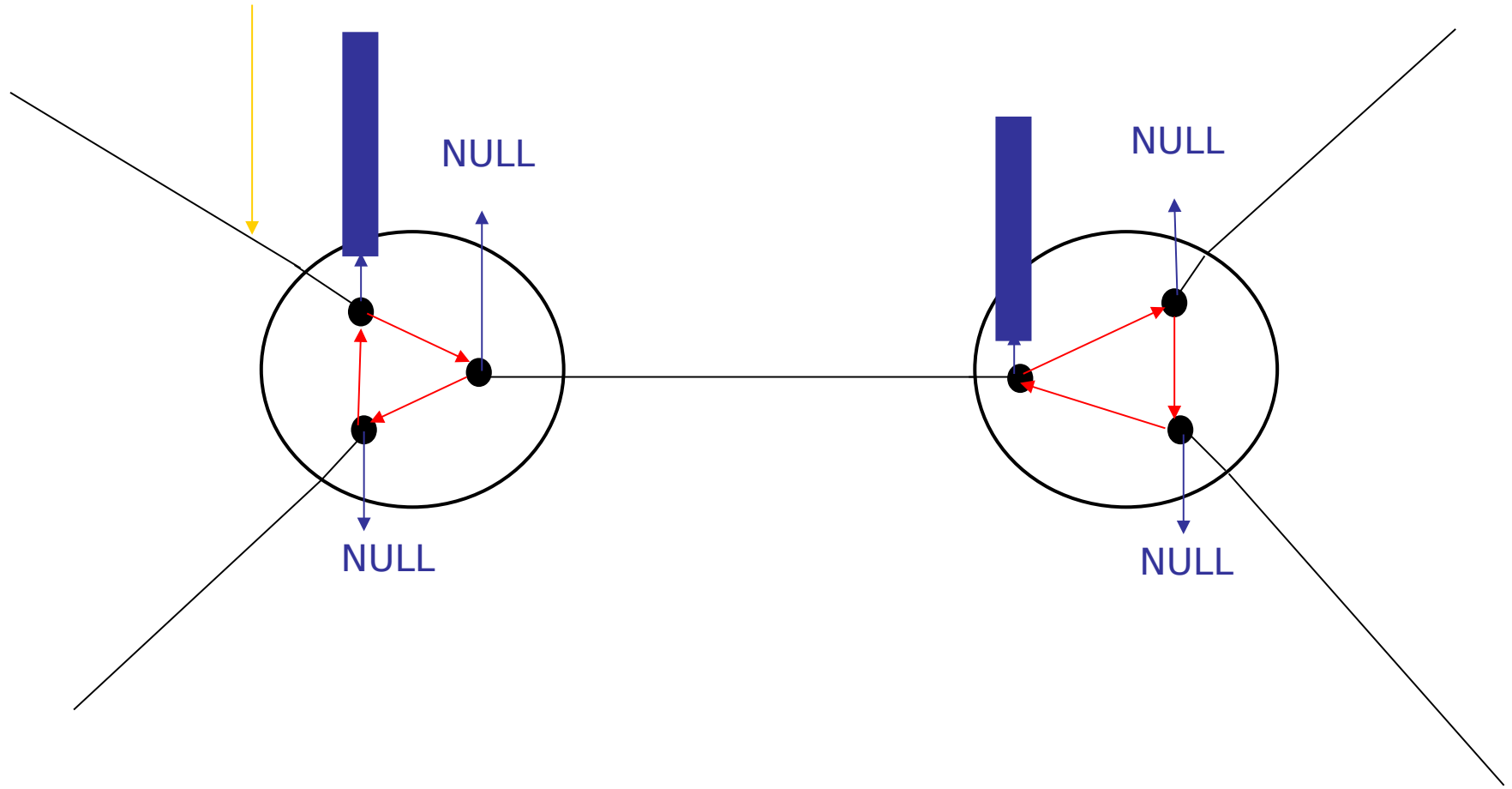
Memory Organization: CLVs with a Rooted View



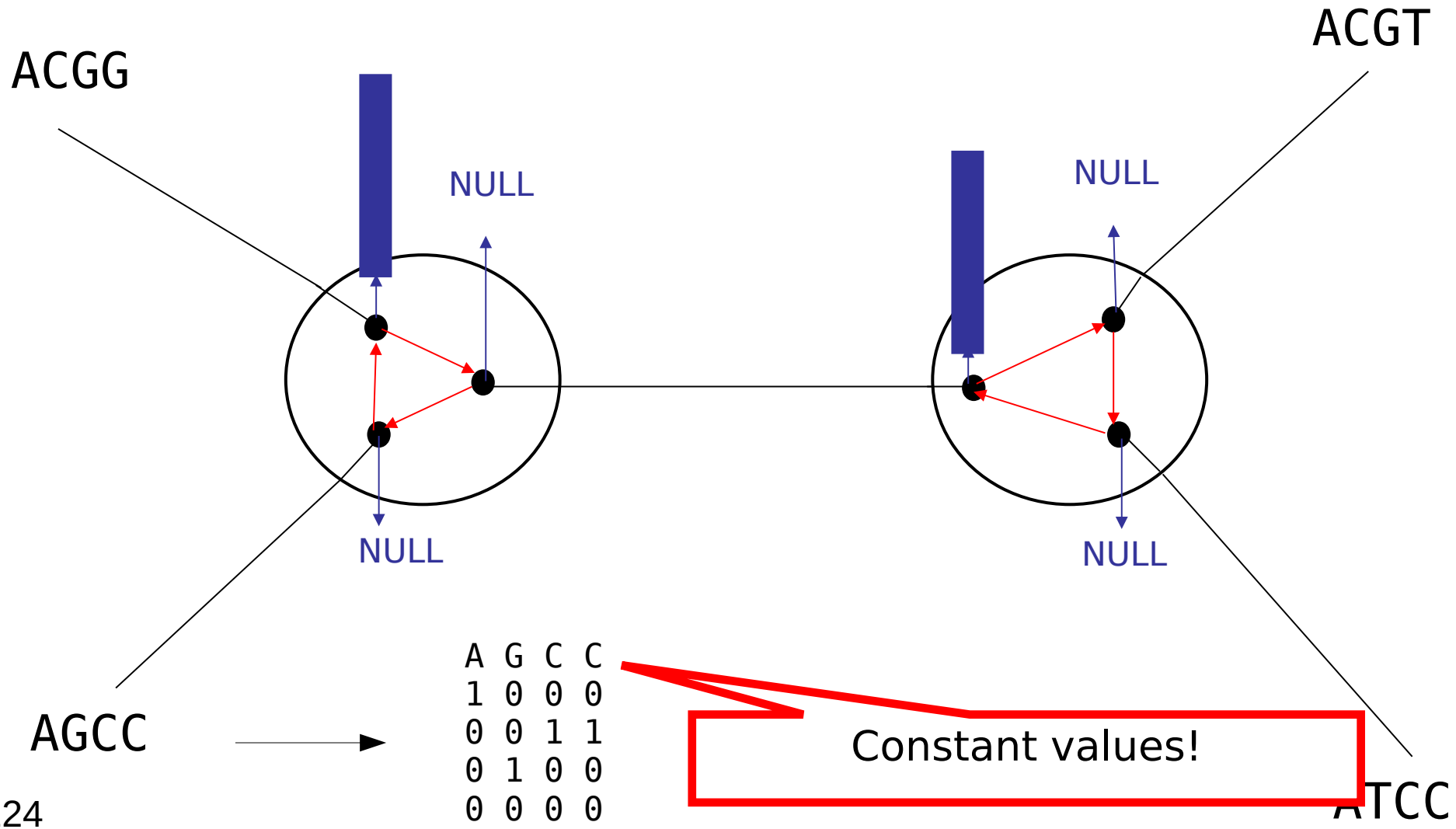
Memory Organization: CLVs with a Rooted View



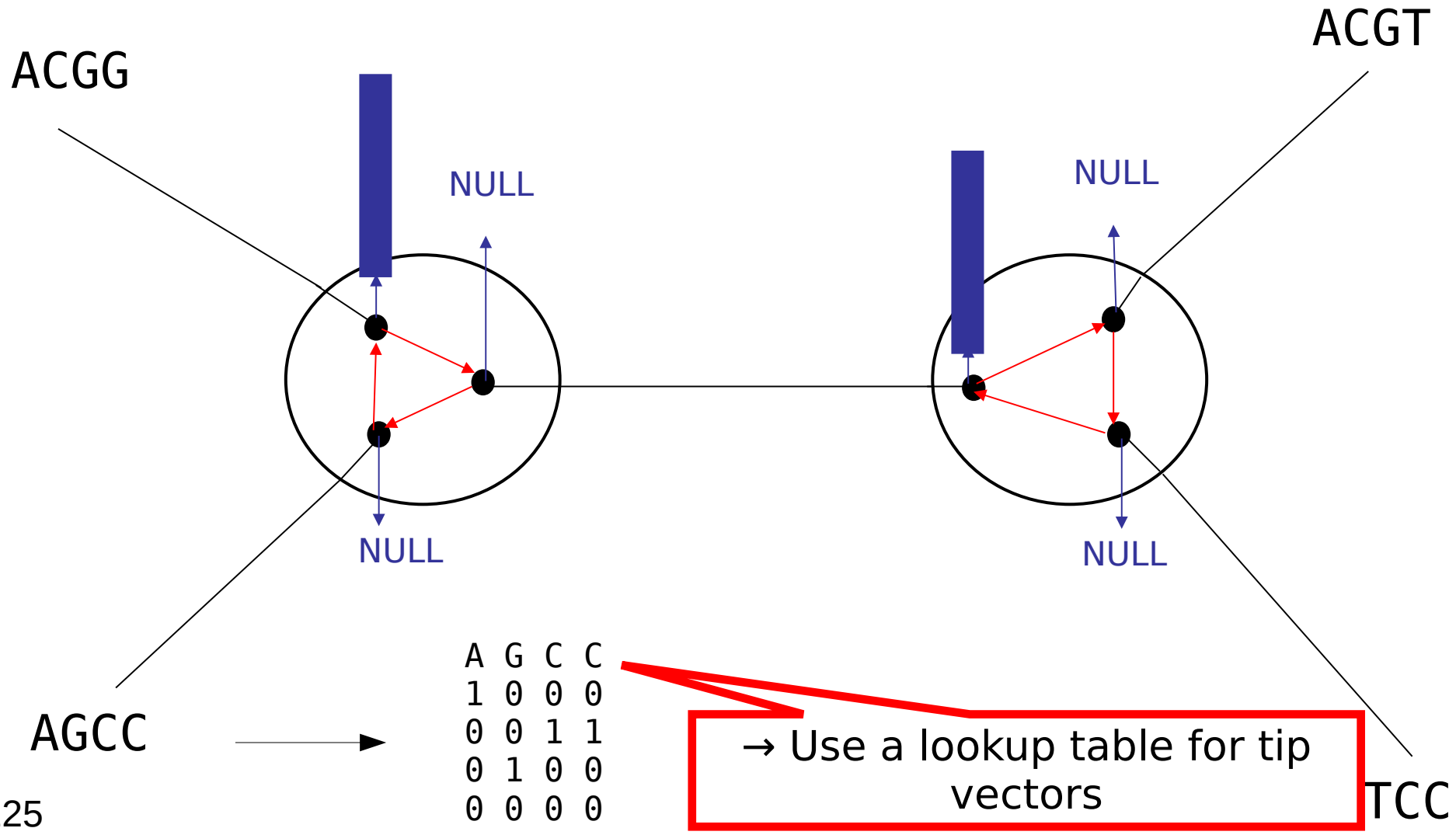
Memory Organization: Ancestral Vectors with a Rooted View



Memory Organization: Tip Vectors



Memory Organization: Tip Vectors



Optimization of Likelihood Calculations

- Use SSE3 & AVX vector intrinsics
- Also: GPUs, FPGAs
- Special implementations (why?) for computing CLVs:

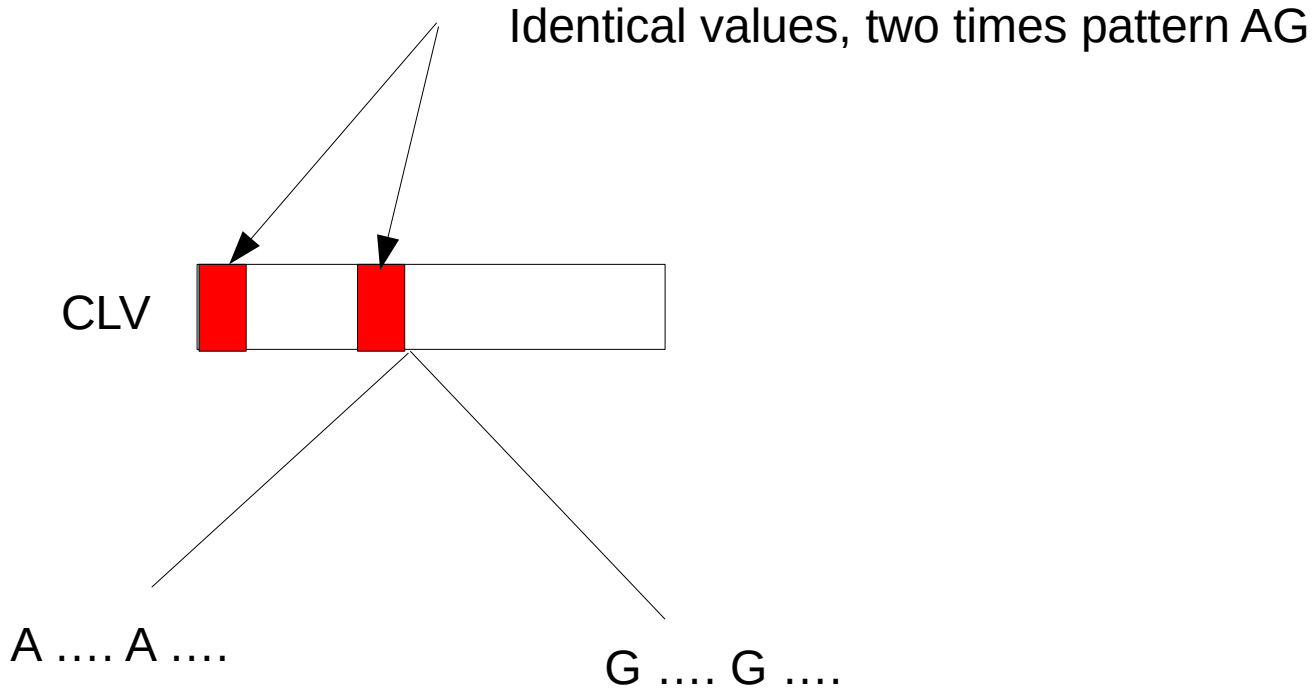
Optimization of Likelihood Calculations

- Use SSE3 & AVX vector intrinsics
- Also: GPUs, FPGAs
- Special implementations (why?) for computing CLVs:

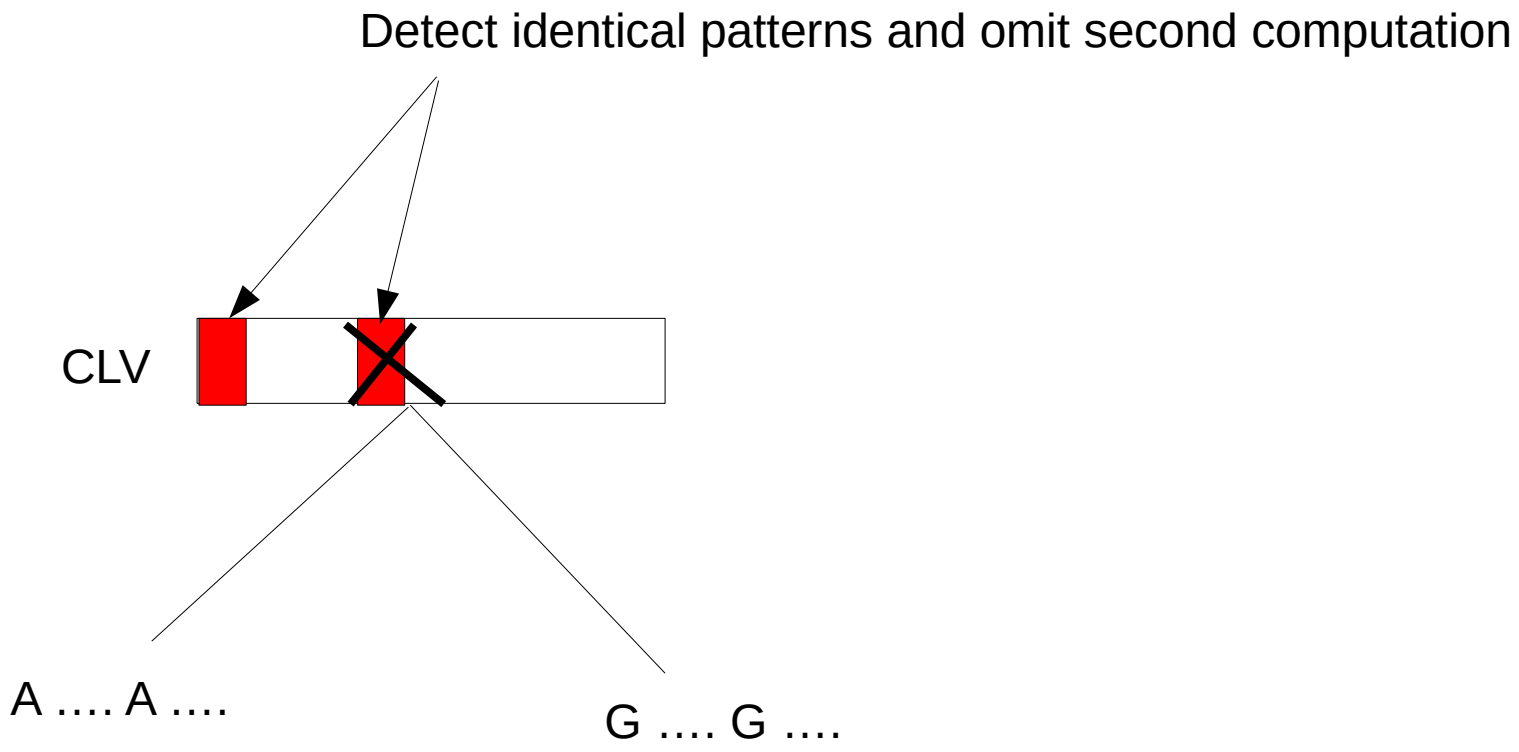
$$\vec{L}_A^{(k)}(c) = \left(\sum_{S=A}^{T} P_{AS}(b_i)\vec{L}_S^{(i)}(c)\right)\left(\sum_{S=A}^{T} P_{AS}(b_j)\vec{L}_S^{(j)}(c)\right)$$

A lot of entries will be zero here if *i* and/or *j* are tips
→ simplify calculations

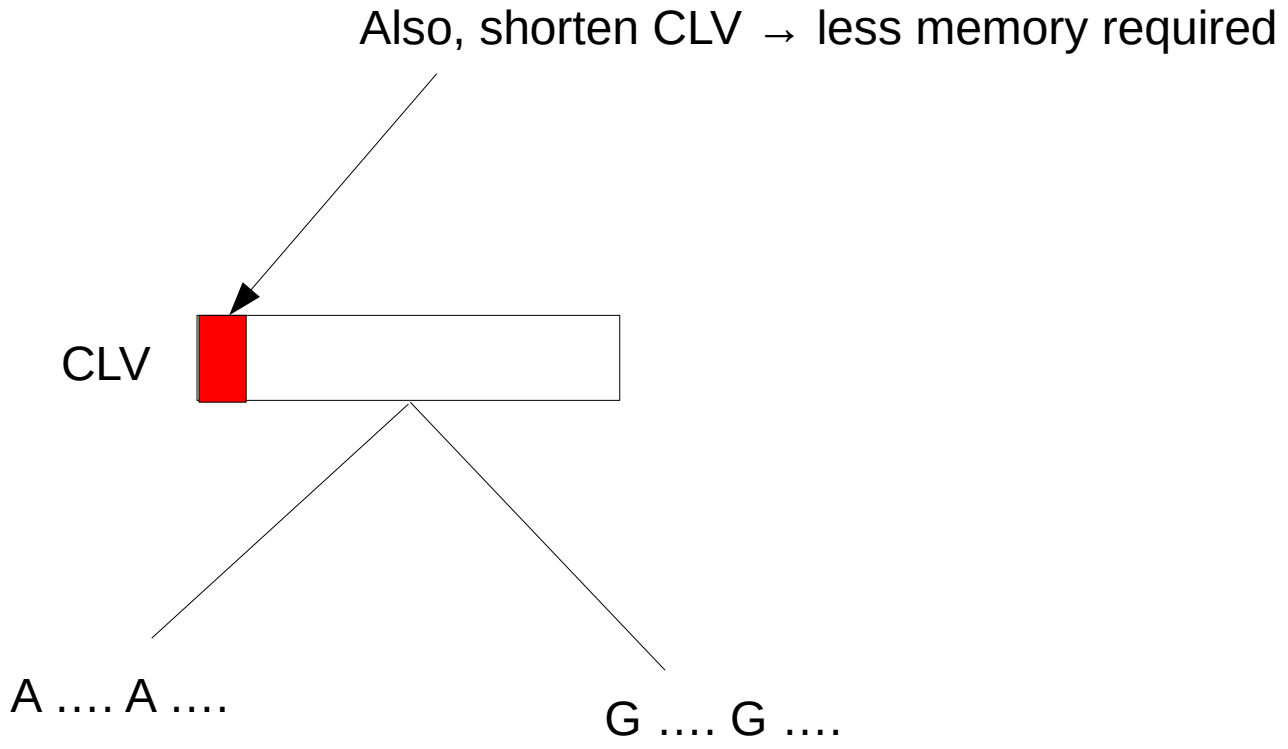
Repeating Patterns



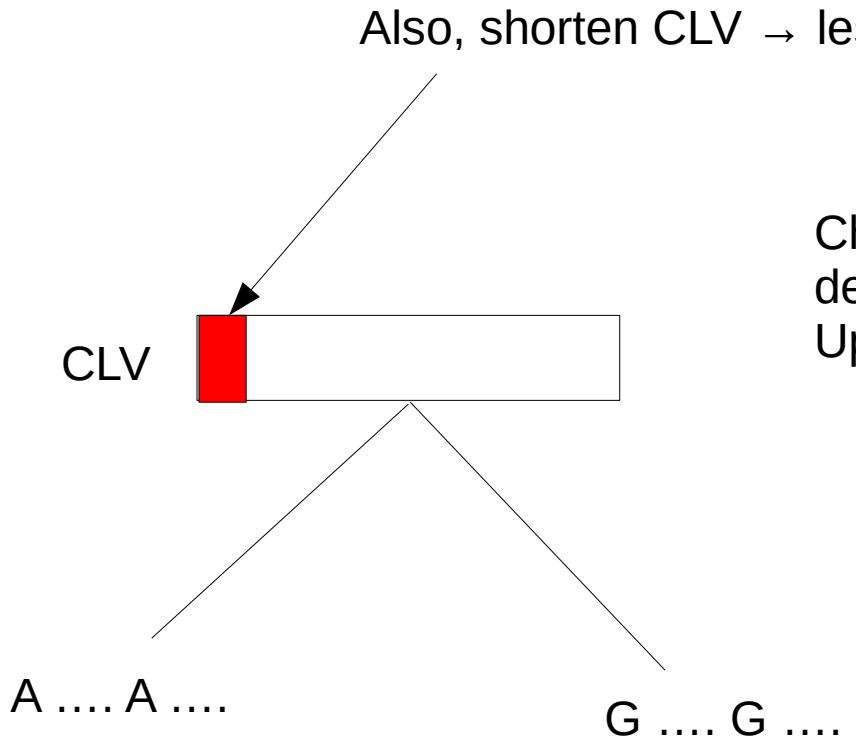
Repeating Patterns



Repeating Patterns



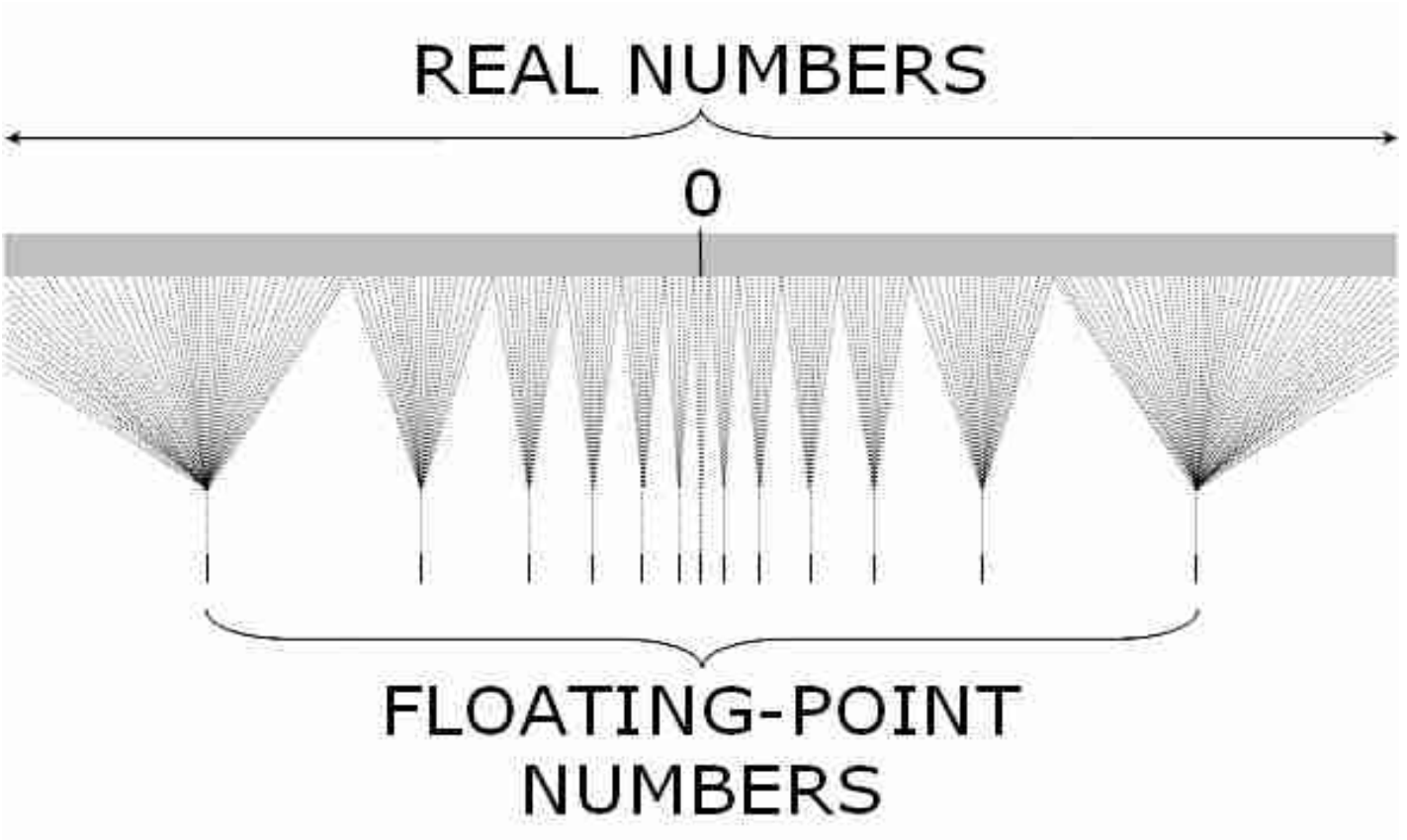
Repeating Patterns (Repeats)



Challenge: Efficient data structure to detect & store repeats
Up to 10-fold run-time improvements

Floating Point Numbers

- Machine numbers are an imperfect mapping of the infinite real numbers to a finite number of machine values!



Floating Point Arithmetics: The Root of All Evil

- Computational science mostly relies on floating-point intensive codes
- How do we verify these codes?
- We stand on shaky grounds
- Scientists using those codes assume that there are no bugs
- Double precision arithmetics required for certain applications
- Who knows what de-normalized floating point numbers are?

→ Please have a look at:

J. Björndalen, O. Anshus: “Trusting floating point benchmarks-are your benchmarks really data-independent?” Applied Parallel Computing. State of the art in Scientific Computing 2010; pp 178-188, Springer.

and at my micro-benchmark at:
https://github.com/stamatak/denormalizedFloatingPointNumbers

Floating Point Arithmetics: The Root of All Evil

- Computational science mostly reli[illegible]e codes
- How do we verify these codes?
- We stand on shaky grounds
- Scientists using those codes assume that there are no bugs
- Double precision arithmetics required for certain applications
- Who knows what de-normalized floating point numbers are?

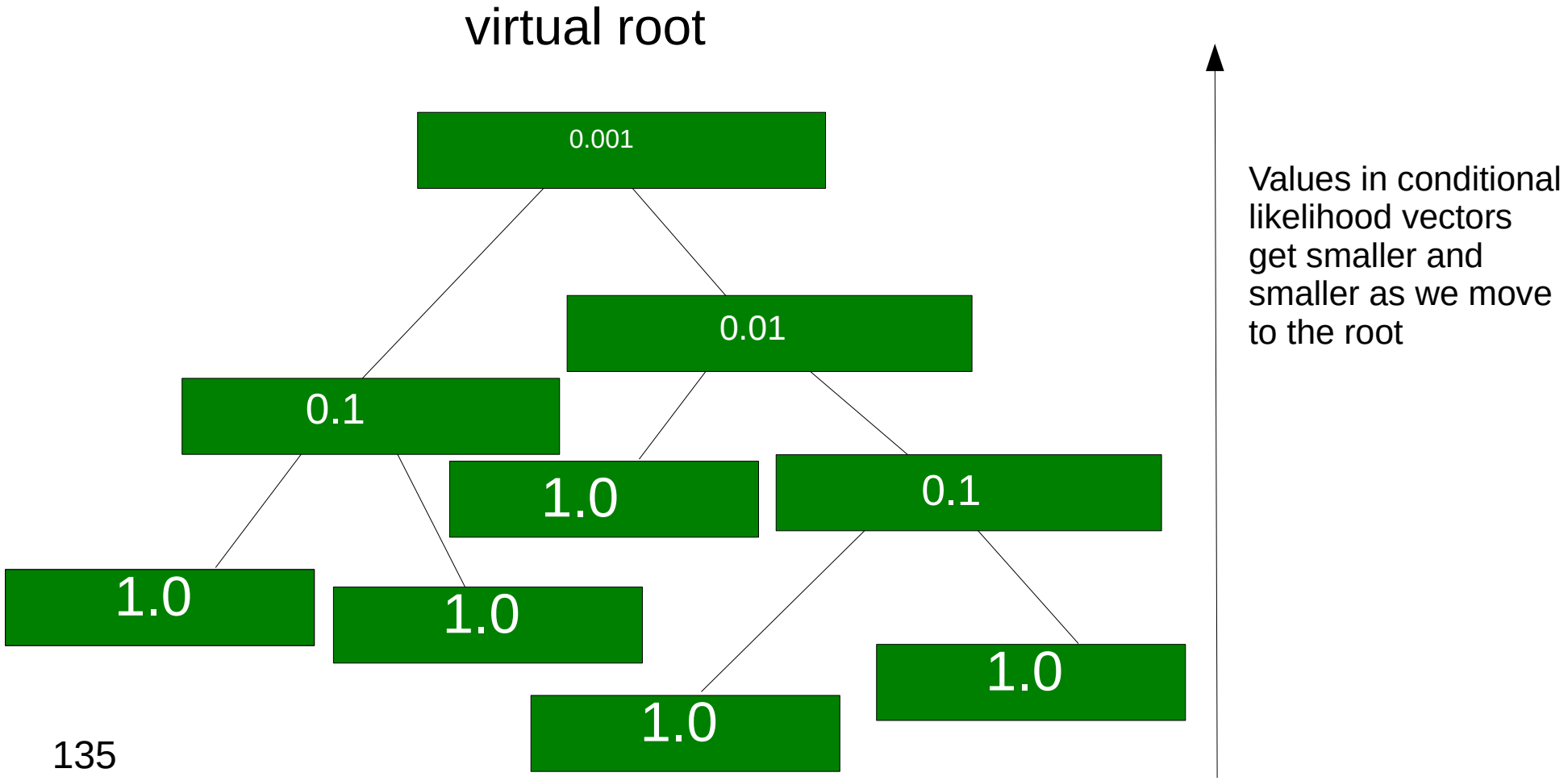
→ Please have a look at:

J. Björndalen, O. Anshus: “Trusting floating point benchmarks-are your benchmarks really data-independent?” Applied Parallel Computing. State of the art in Scientific Computing 2010; pp 178-188, Springer.

and at my micro-benchmark at:
https://github.com/stamatak/denormalizedFloatingPointNumbers

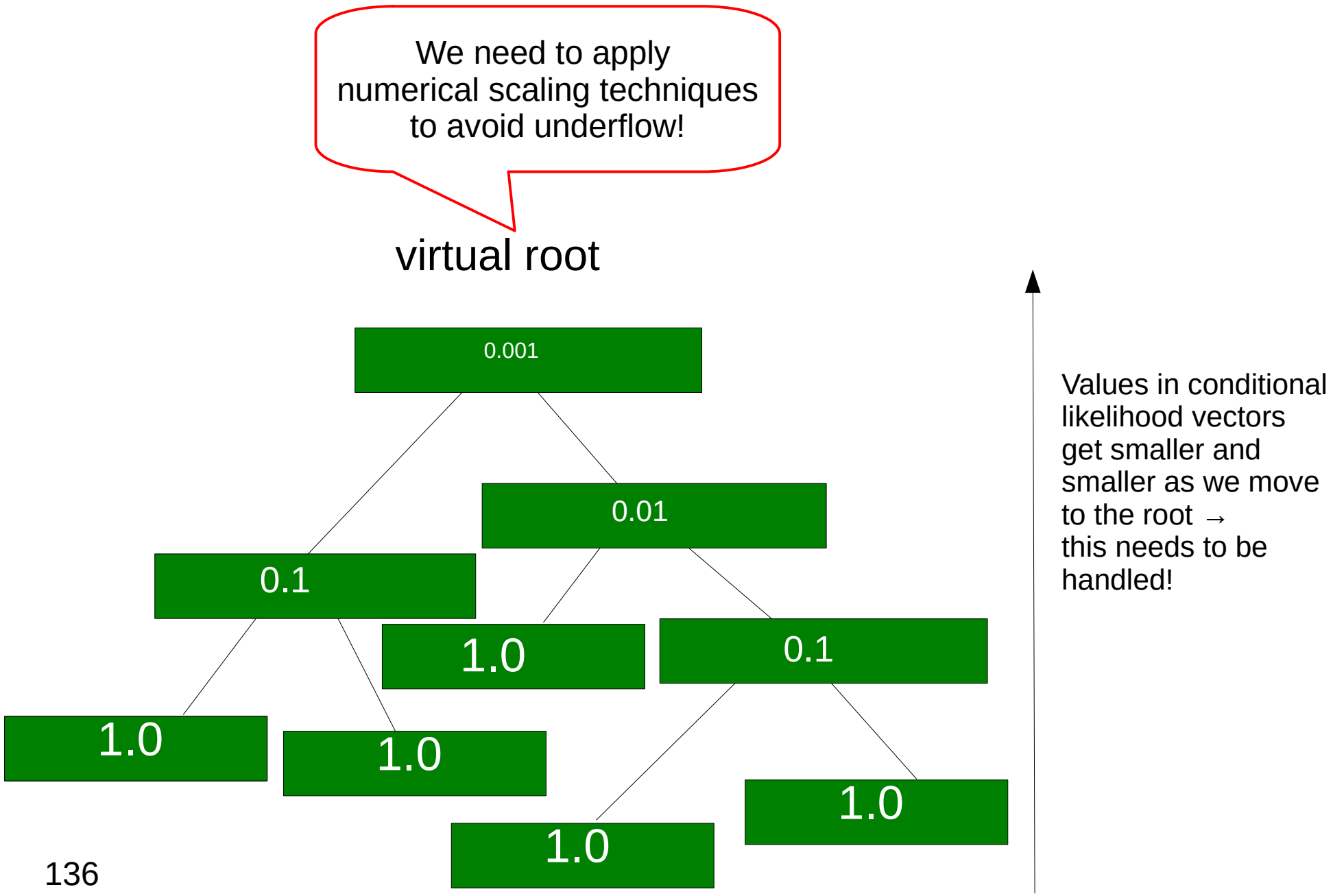
Why is this relevant when talking about Maximum Likelihood?

Post-order Traversal



Post-order Traversal

We need to apply numerical scaling techniques to avoid underflow!

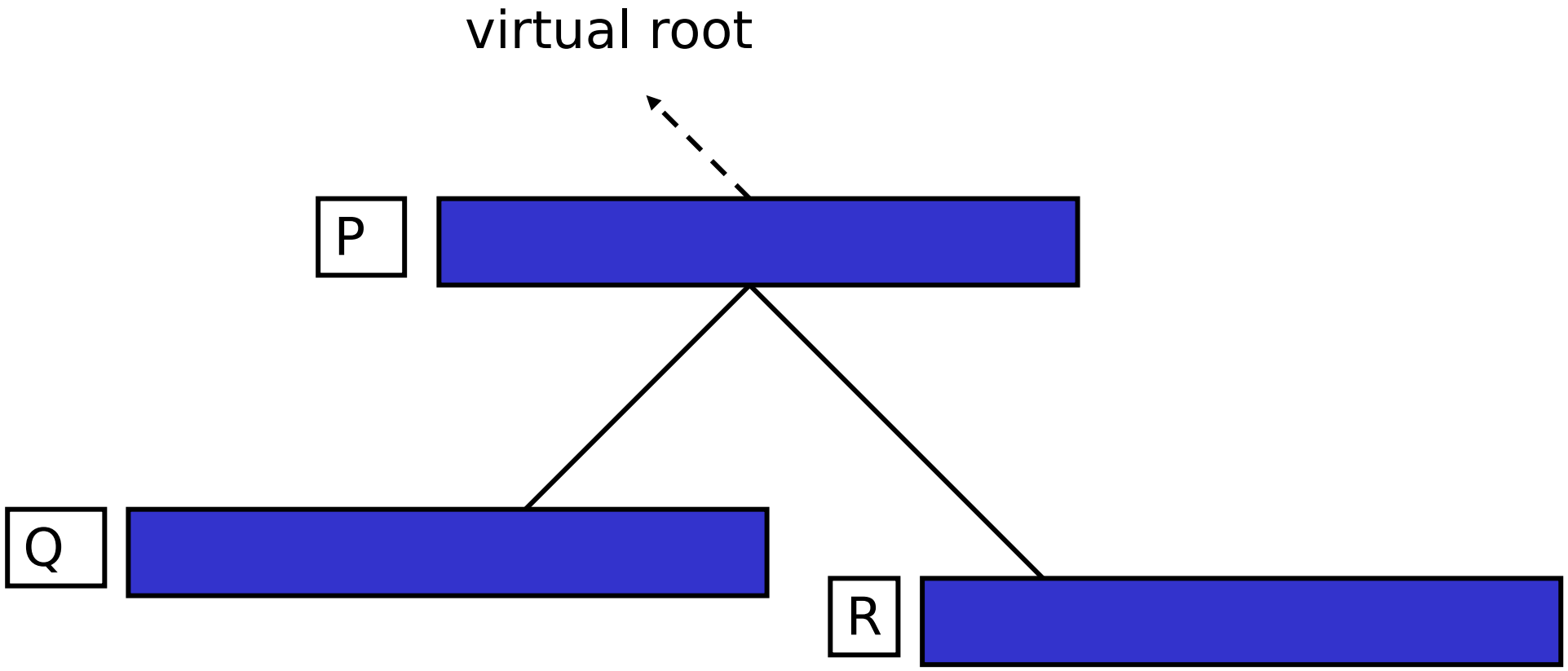


Values in conditional likelihood vectors get smaller and smaller as we move to the root → this needs to be handled!

Outline

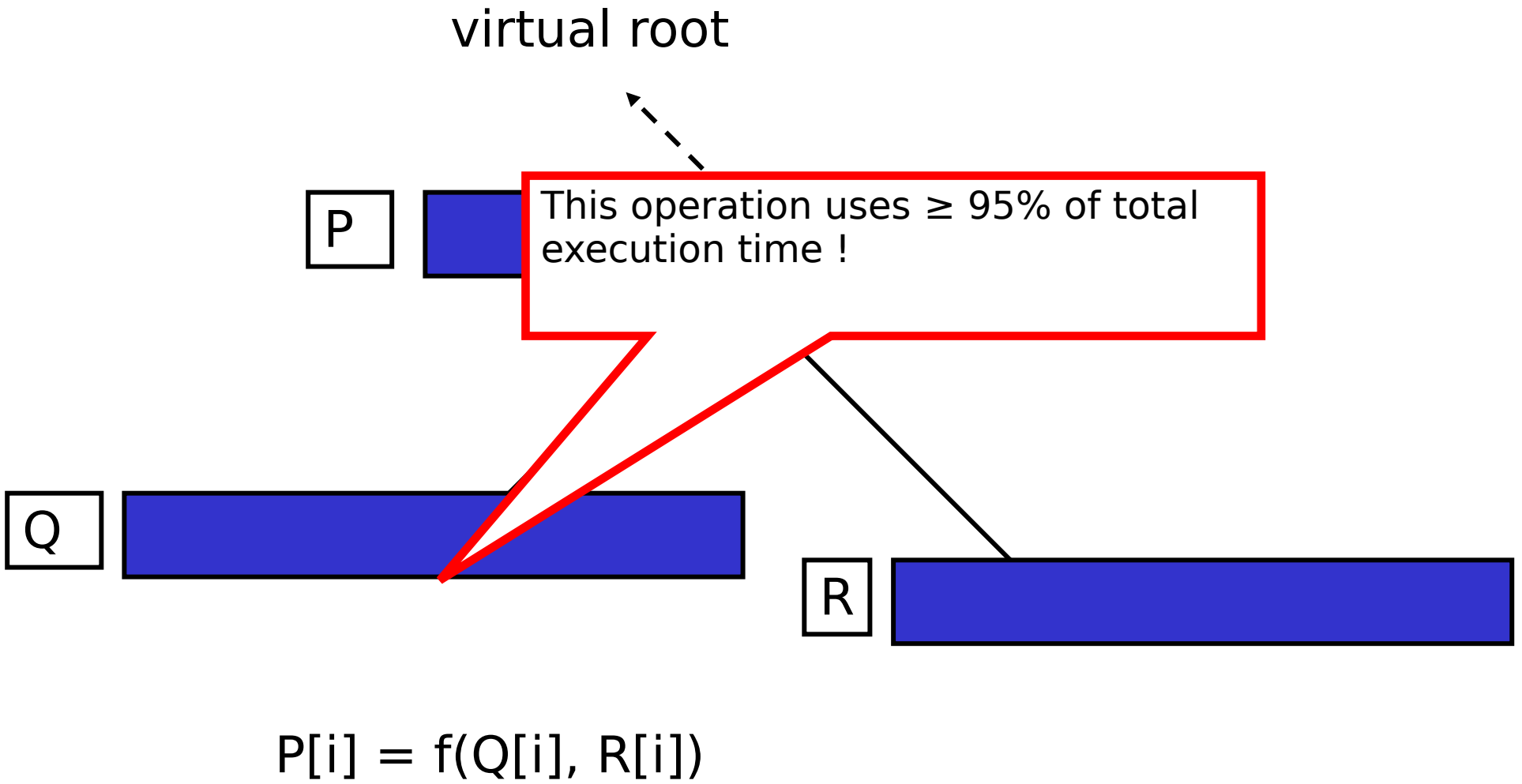
- Last time:
  - How to Compute the Likelihood of a tree
  - How to compute the Likelihood efficiently: Felsenstein Pruning Algorithm
- Today & next time
  - What is hidden in P(t) – what do the models look like ?
  - How to compute the Maximum Likelihood score on a tree?
  - Advanced substitution models
  - Efficiently computing the Likelihood on trees
  - **Parallel Likelihood computations**

Loop Level Parallelism



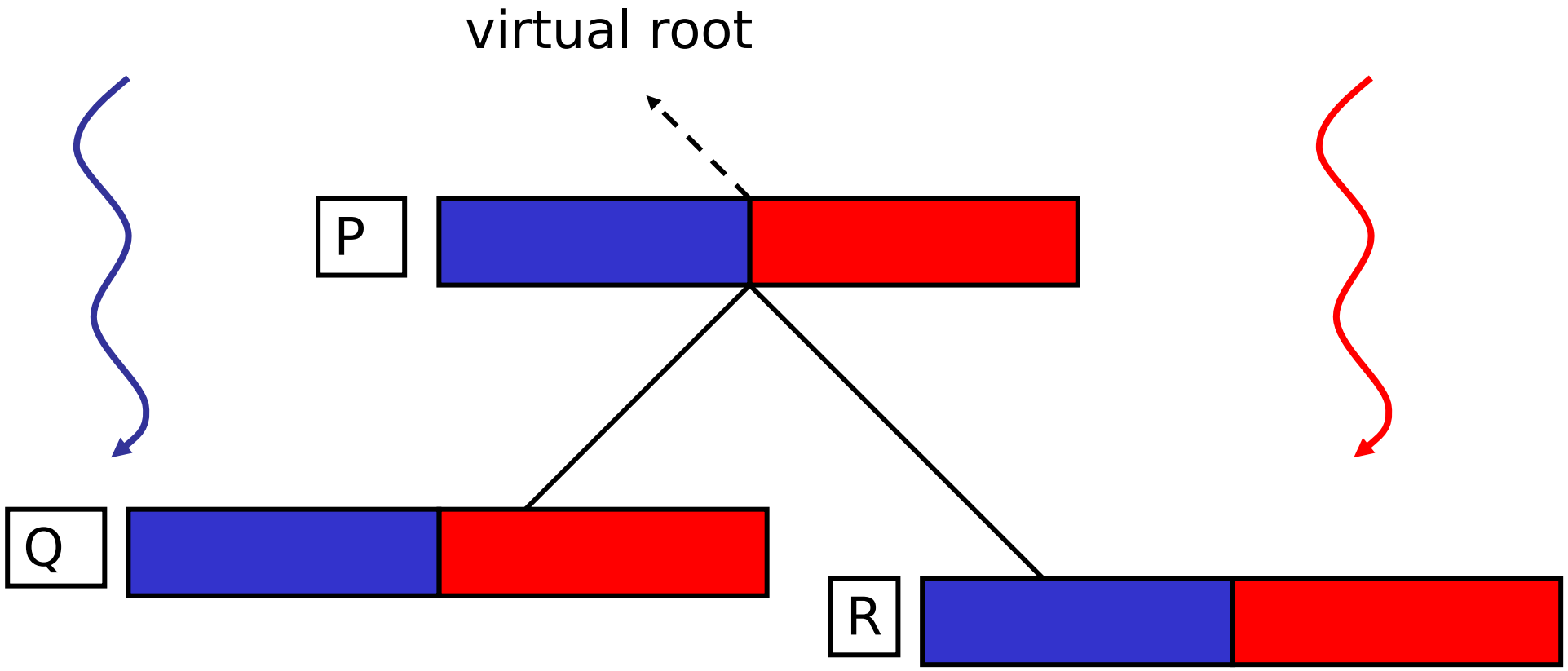
P[i] = f(Q[i], R[i])

Loop Level Parallelism

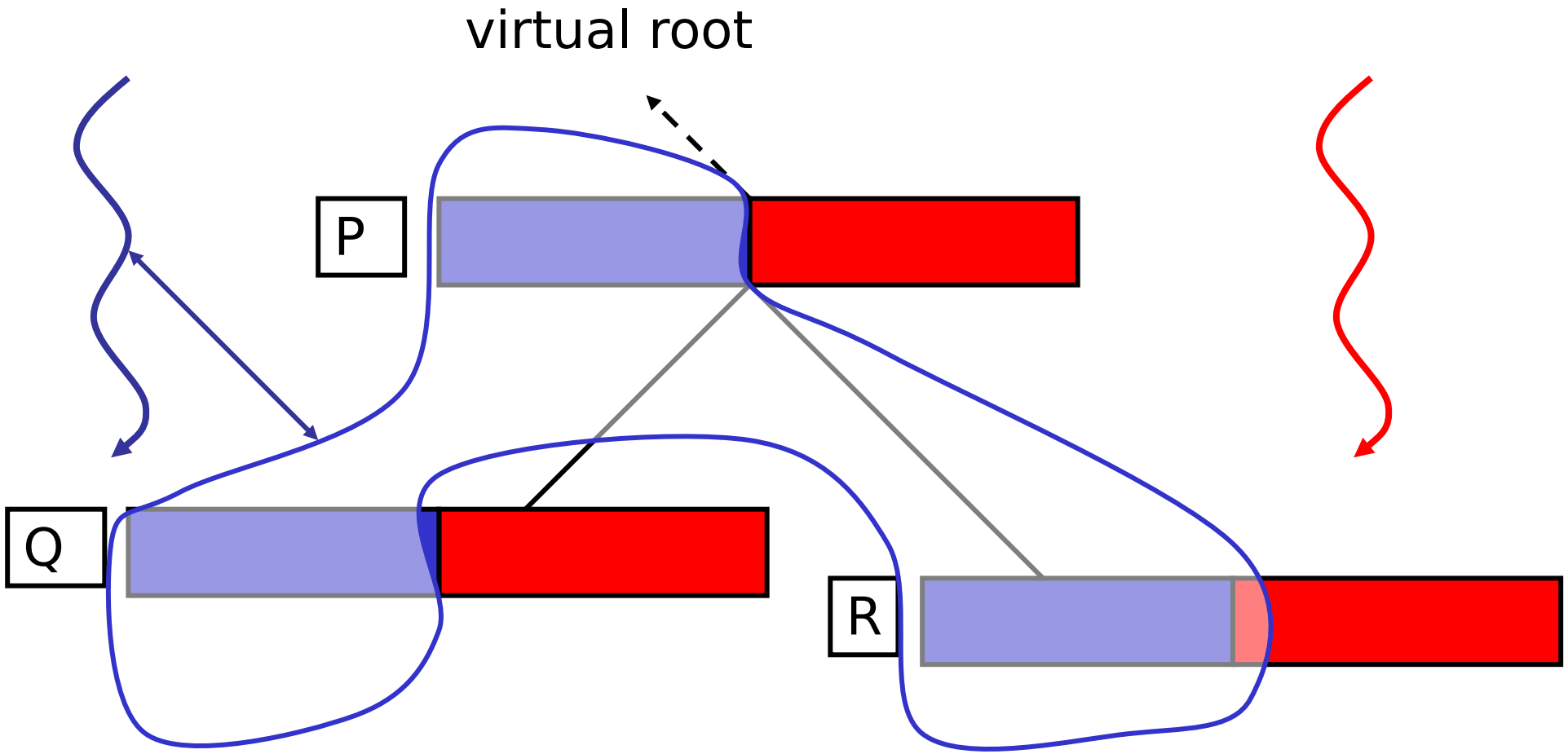


P[i] = f(Q[i], R[i])

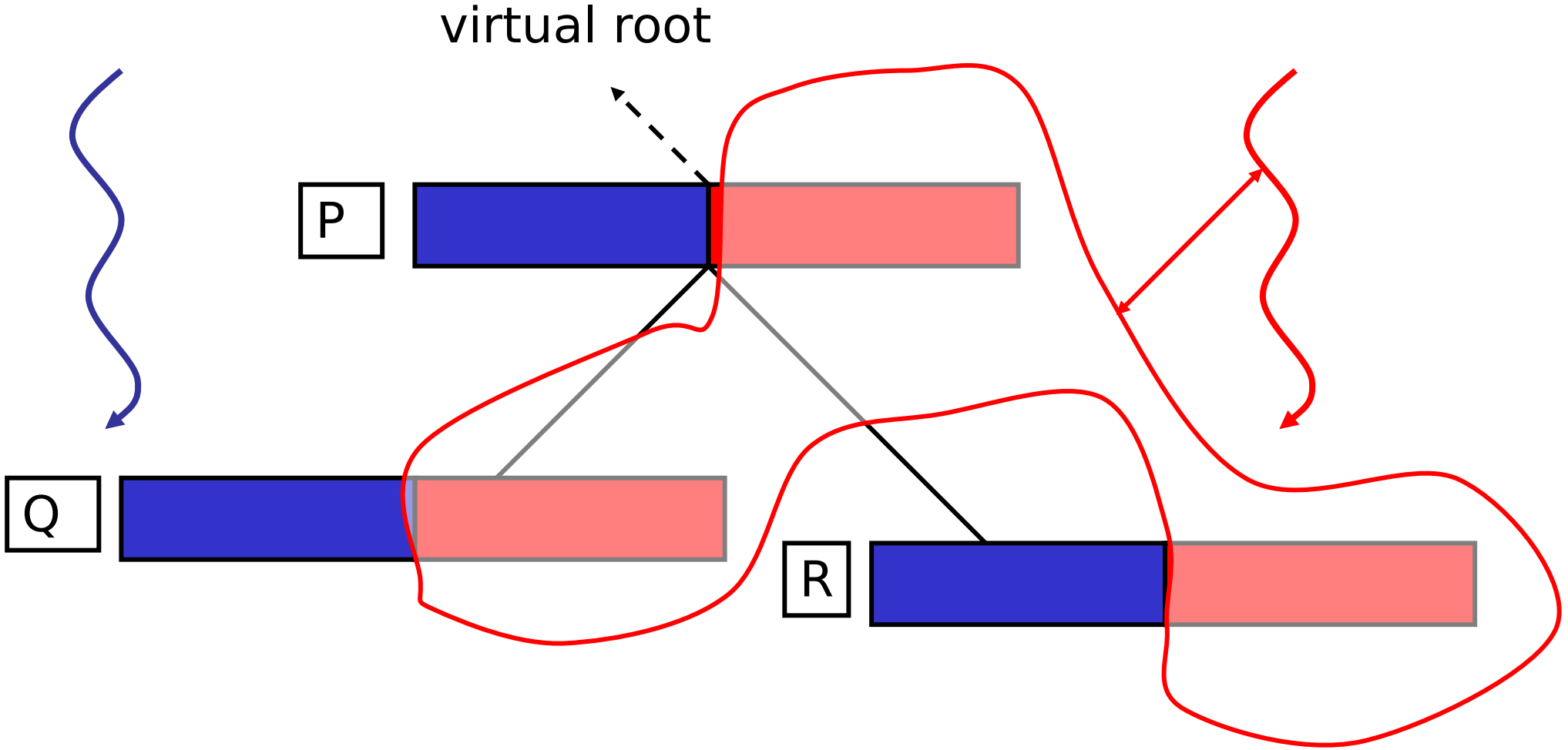
Loop Level Parallelism



Loop Level Parallelism



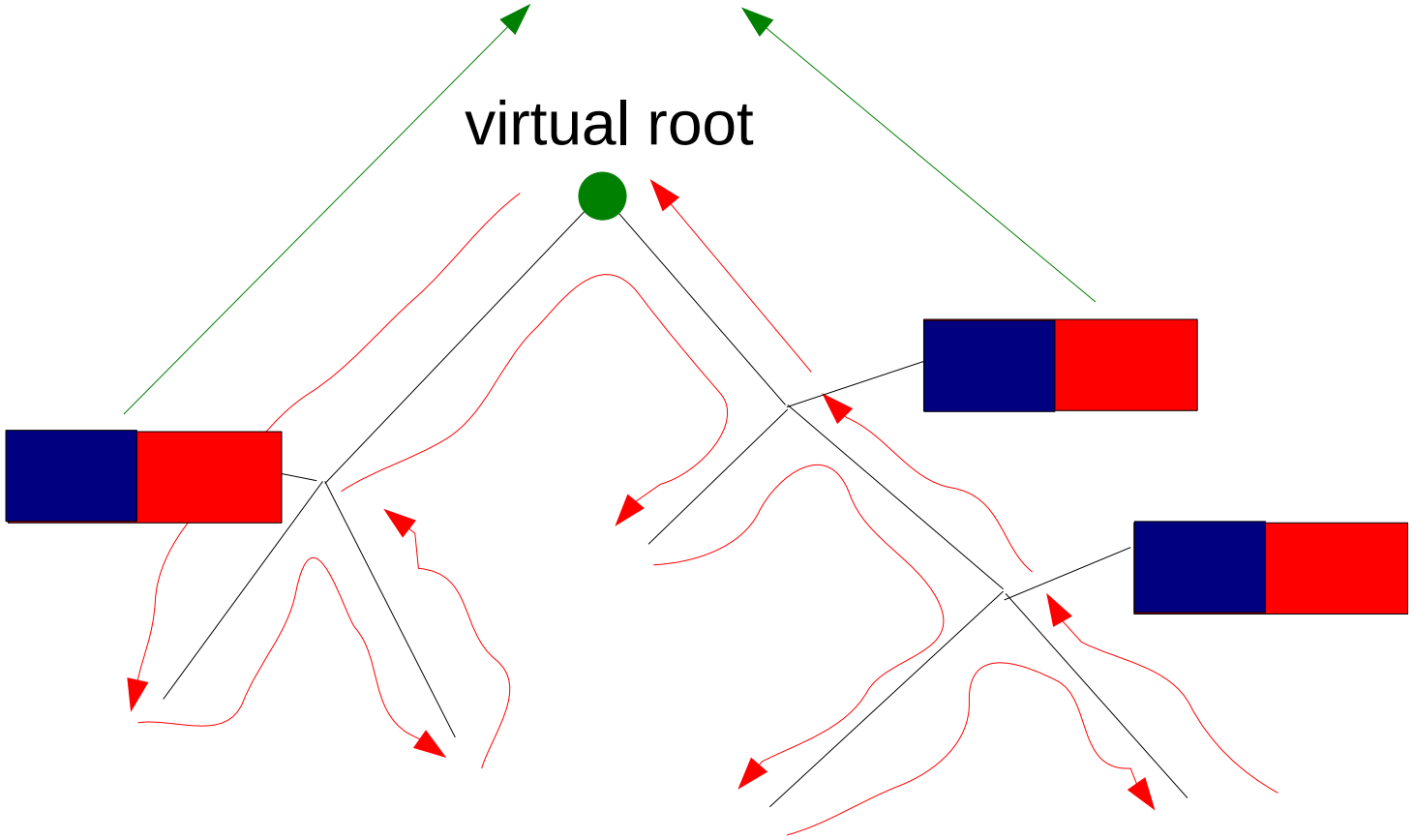
Loop Level Parallelism



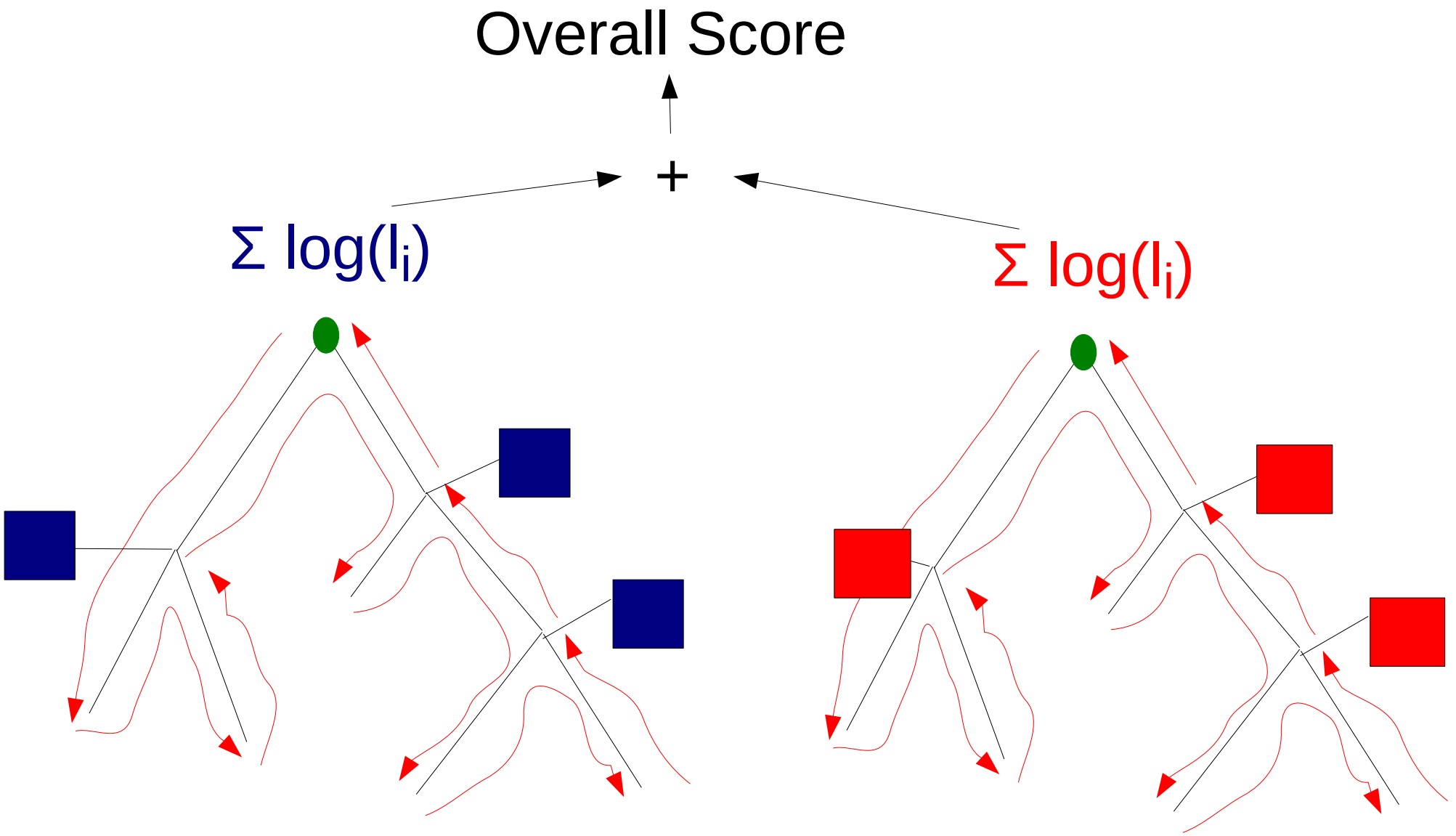
Parallel Post-order Traversal

Only need to synchronize at the root
→ MPI_Reduce() to calculate:

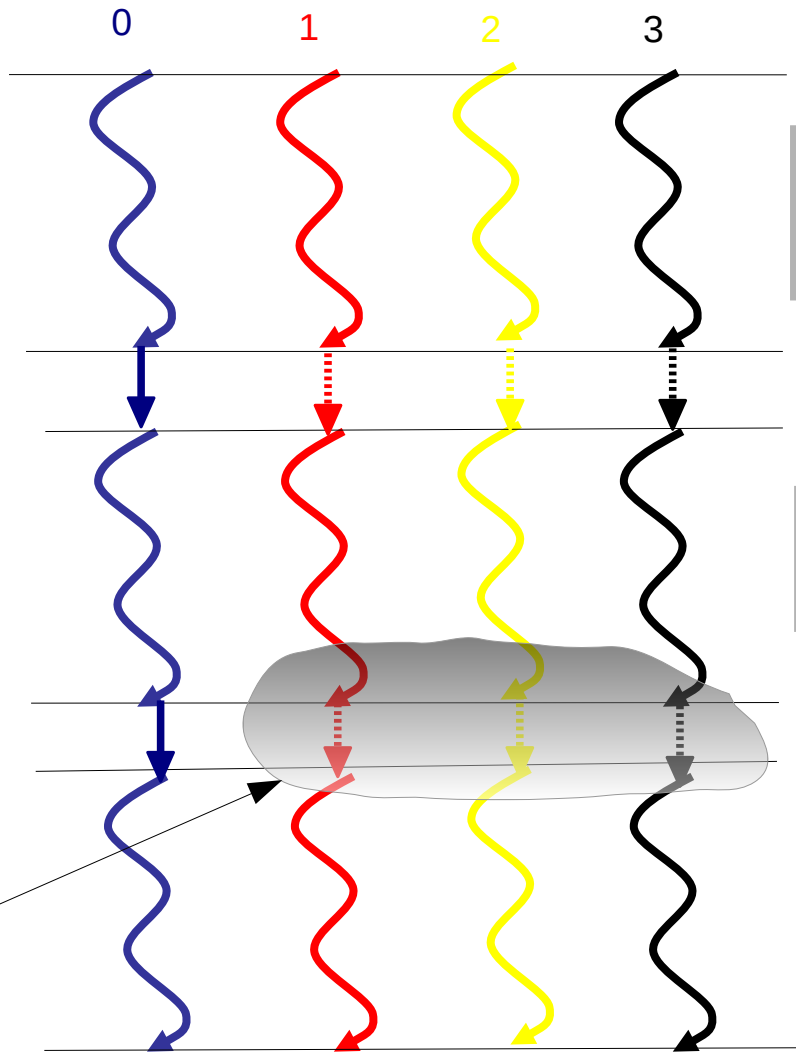
$\Sigma \log(l_i)$



Parallel Post-order Traversal



Classic Fork-Join with Busy-Wait



Broadcast Traversal — Trigger

Compute all vectors in Traversal descriptor

Reduce $\Sigma \log(l_i)$ — Barrier

Broadcast Traversal — Trigger

Compute all vectors in Traversal descriptor

Reduce $\Sigma \log(l_i)$ — Barrier

Broadcast new α params. — Trigger

Compute all vectors in Traversal descriptor

Reduce $\Sigma \log(l_i)$ — Barrier

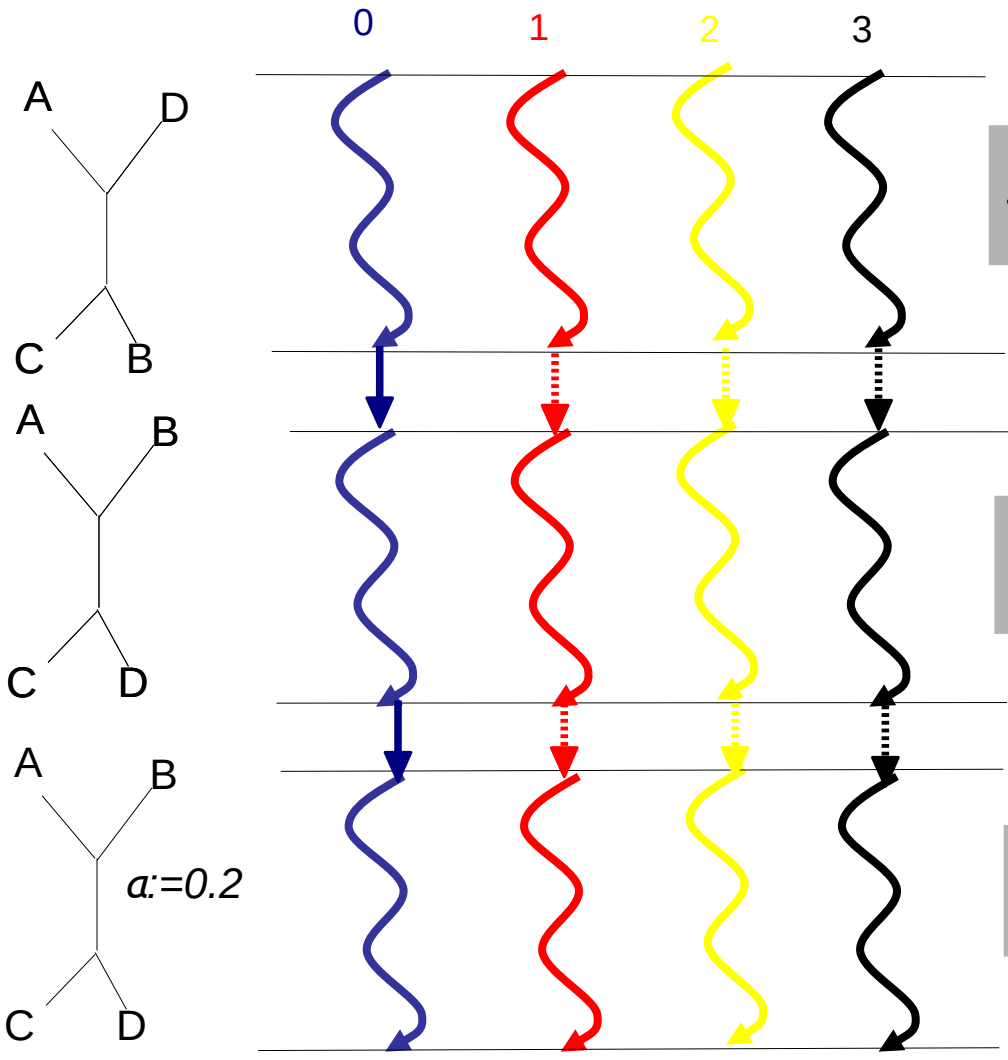
Busy wait

Synchronizations in RAxML with Pthreads

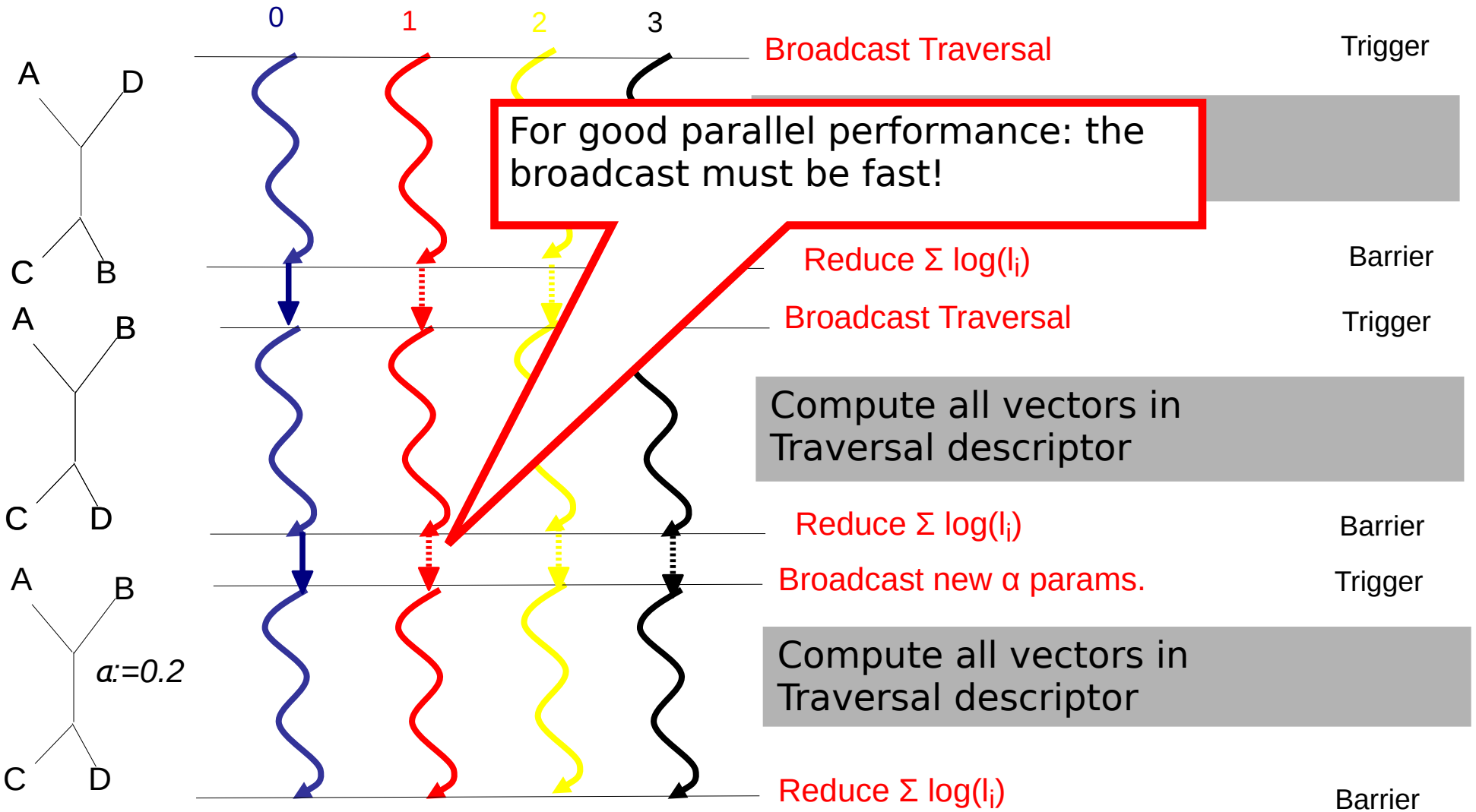
- RAxML Pthreads for a run time of about **10 seconds** on 16 cores/threads
- 404 taxa 7429 sites: **194,000** Barriers
- 1481 taxa 1241 sites: **739,000** Barriers
- A paper on performance of alternative PThreads barrier implementations:

  S.A. Berger, A. Stamatakis: "Assessment of Barrier Implementions for Fine-Grain Parallel Regions on Current Multi-core Architectures", *IEEE Cluster* 2010.

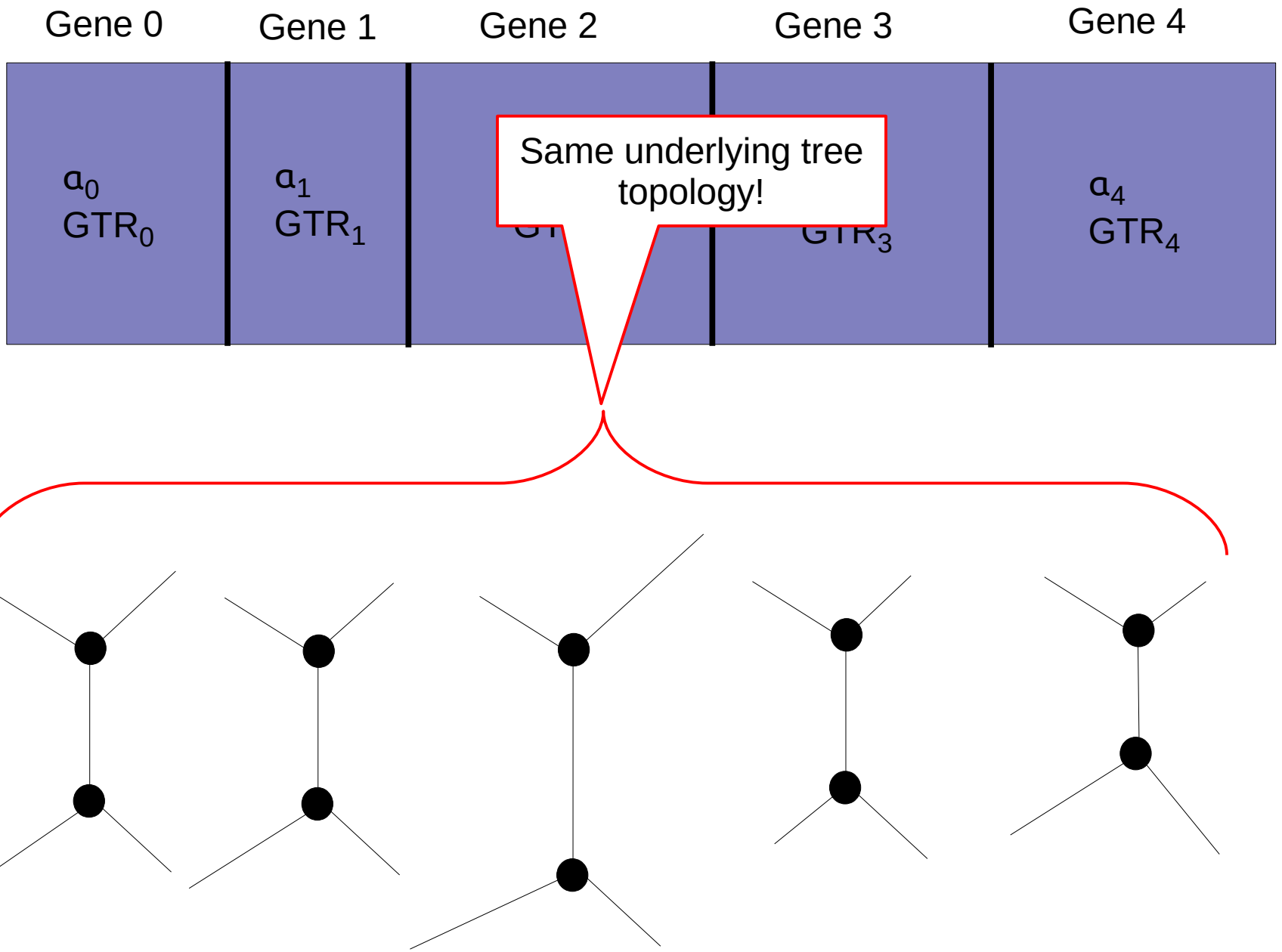
Classic Fork-Join with Busy-Wait (model optimization)



Classic Fork-Join with Busy-Wait (model optimization)



Problems start with partitioned datasets!

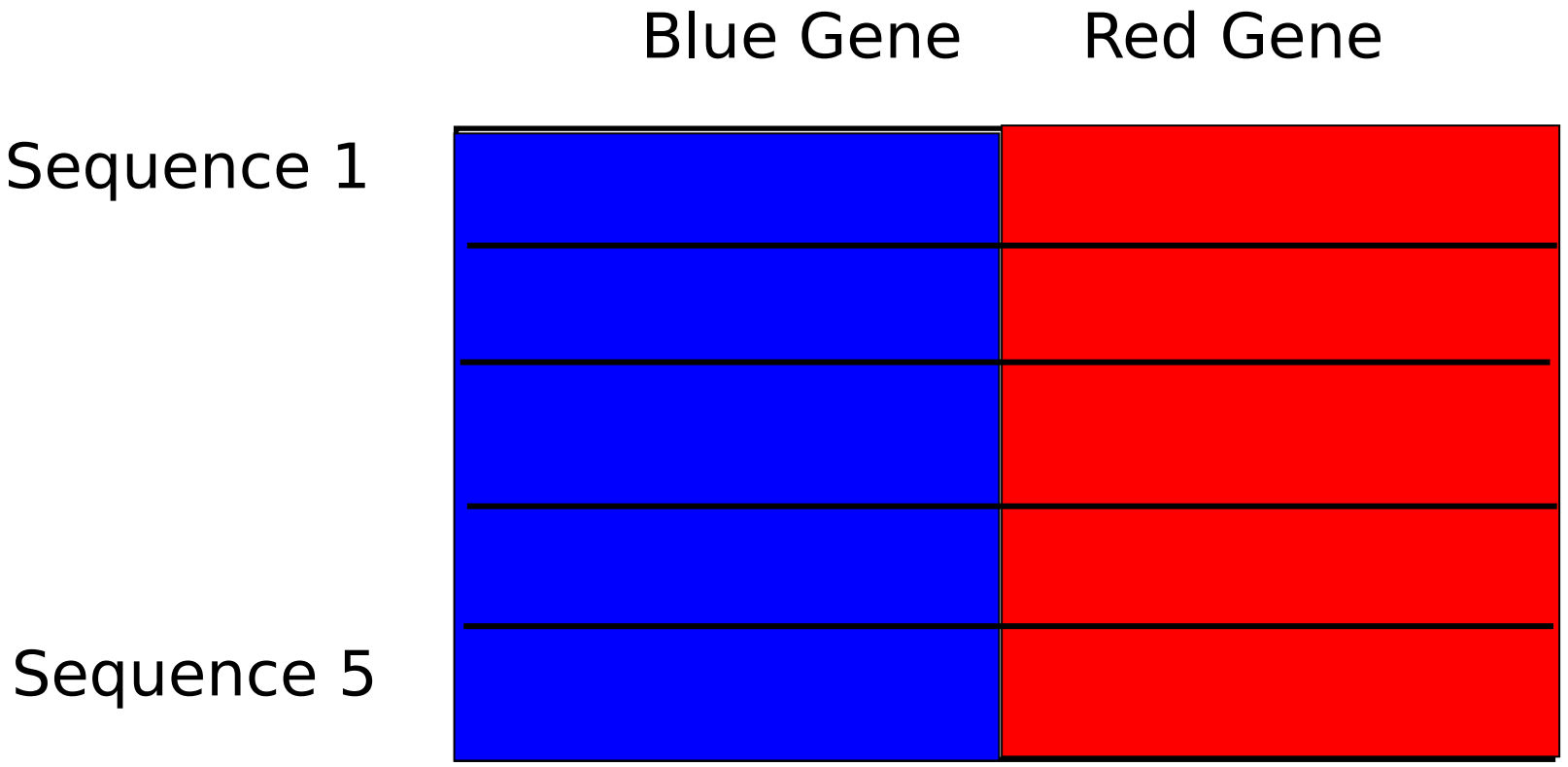


Parallel Performance Problems

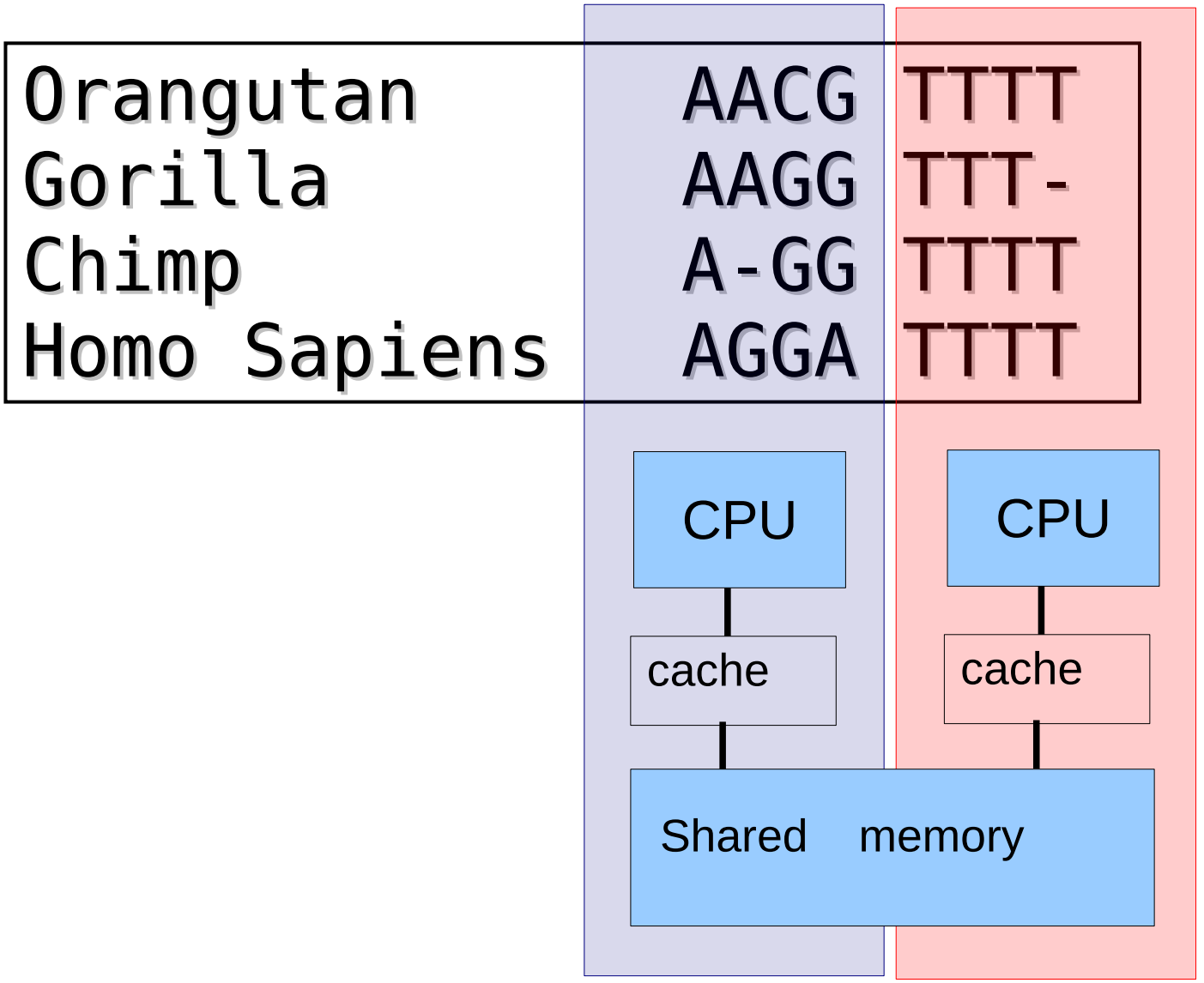
- They all start with partitioned datasets!
- How do we distribute partitions to processors?
- How do we calculate parameter changes?
- How much time does our broadcast take?
- Goal: Keep all processors busy all the time

  → minimize communication and synchronization!

Example

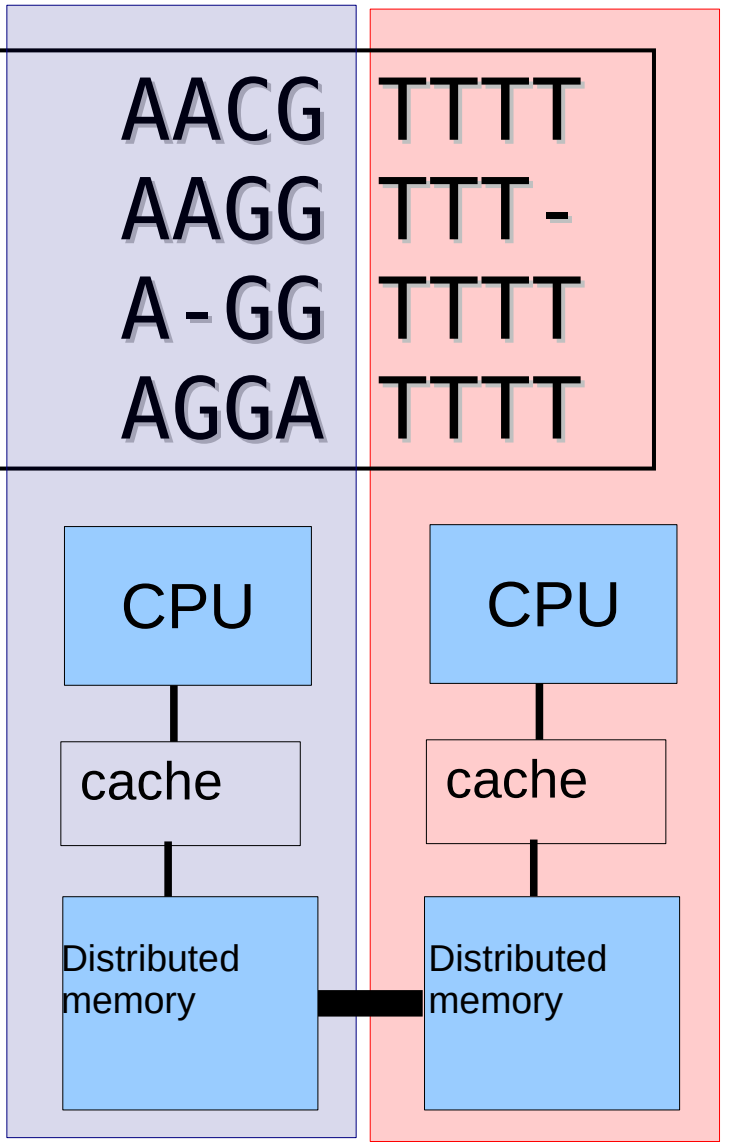


Data Distribution



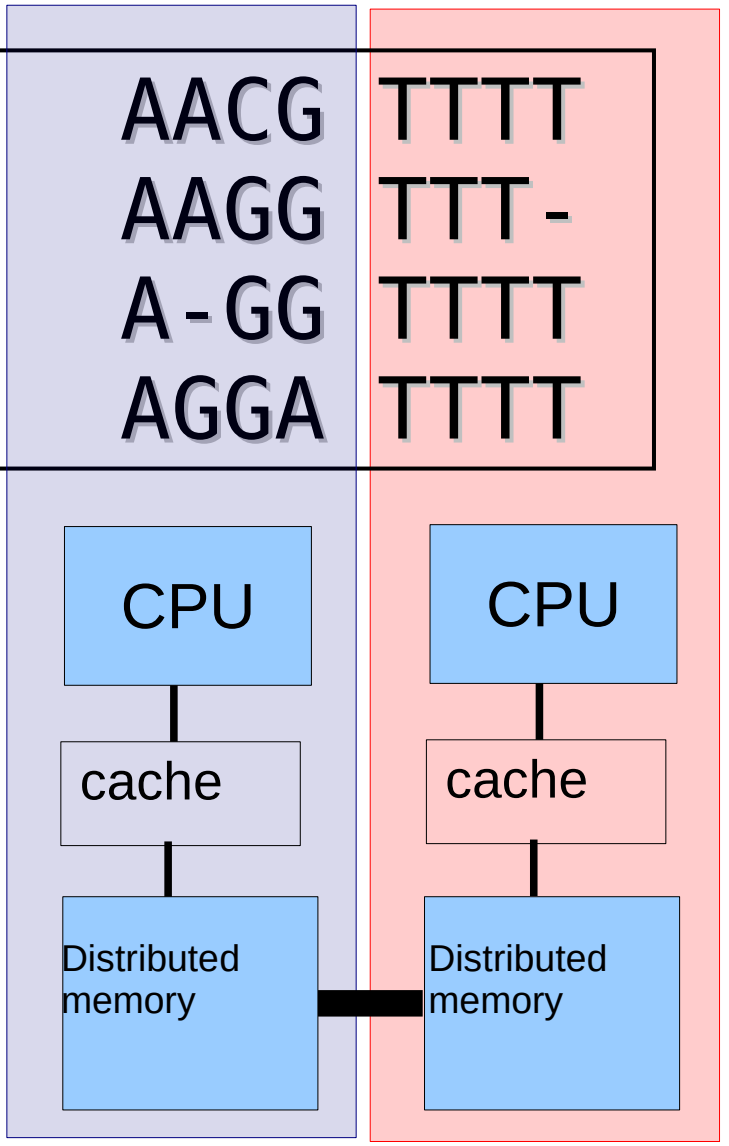
Data Distribution

```
Orangutan      AACG TTTT
Gorilla        AAGG TTT-
Chimp          A-GG TTTT
Homo Sapiens   AGGA TTTT
```



Data Distribution

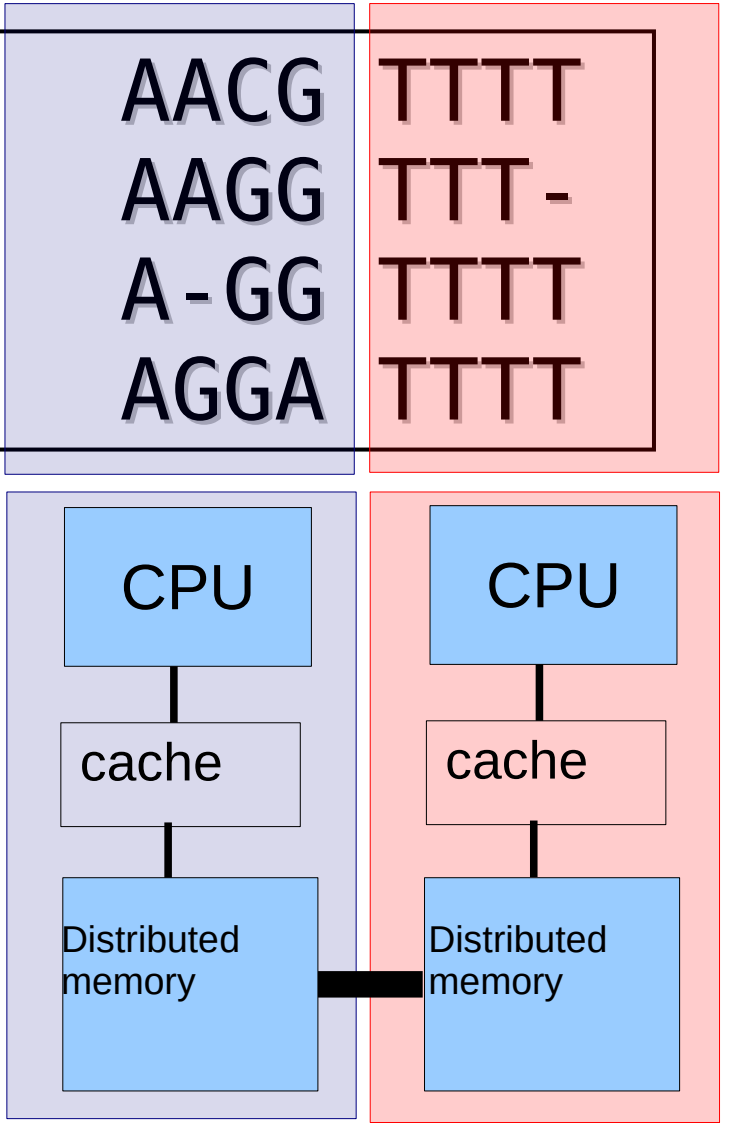
```
Orangutan      AACG TTTT
Gorilla        AAGG TTT-
Chimp          A-GG TTTT
Homo Sapiens   AGGA TTTT
```



Partitioned data distribution is not that trivial!

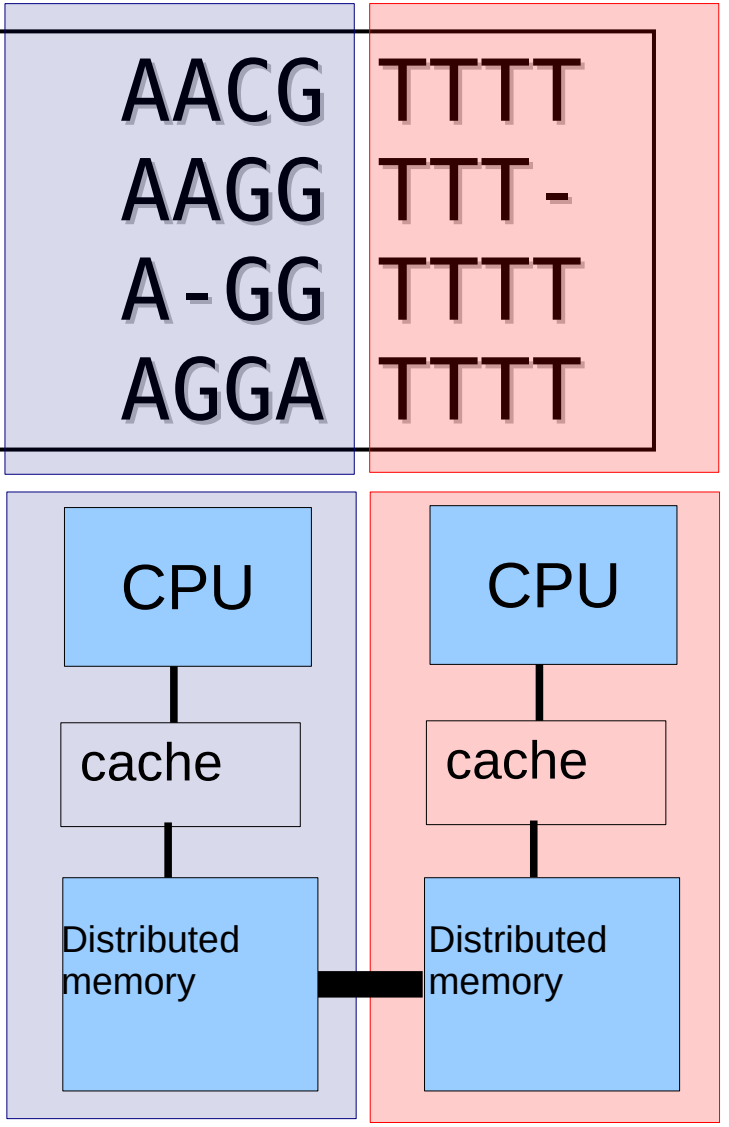
Data Distribution I

```
Orangutan      AACG TTTT
Gorilla        AAGG TTT-
Chimp          A-GG TTTT
Homo Sapiens   AGGA TTTT
```



Data Distribution I

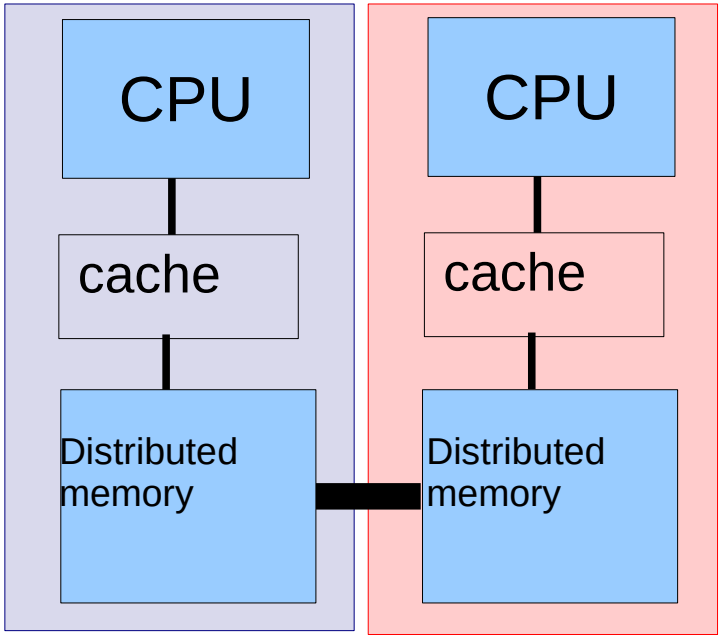
```
Orangutan      AACG TTTT
Gorilla        AAGG TTT-
Chimp          A-GG TTTT
Homo Sapiens   AGGA TTTT
```



Works well when we have more partitions than processors:
May lead to load imbalance not all processors obtain equal number of sites!
Not all partitions have equal length!

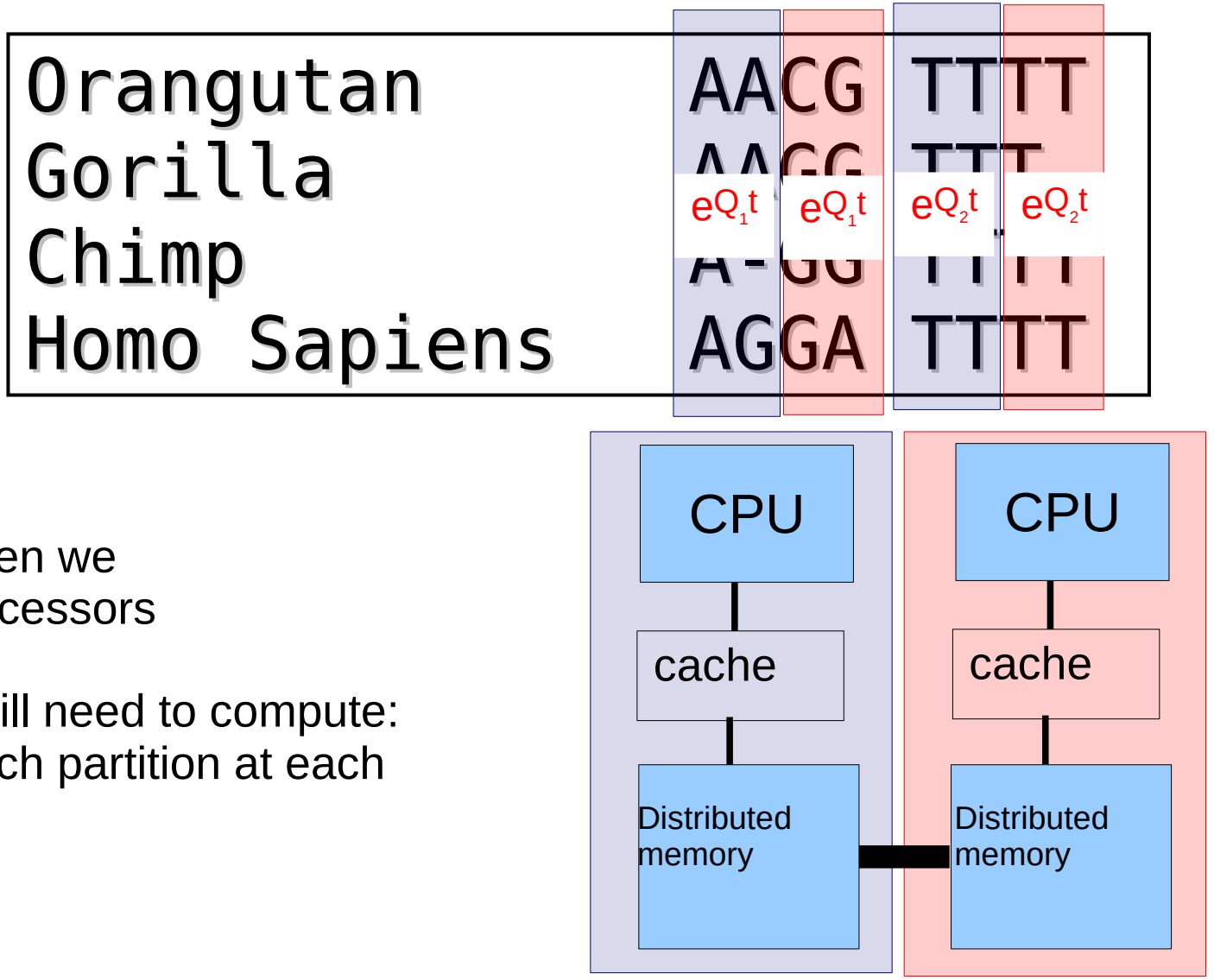
Data Distribution II

```
Orangutan      AACG TTTT
Gorilla        AAGG TTT-
Chimp          A-GG TTTT
Homo Sapiens   AGGA TTTT
```



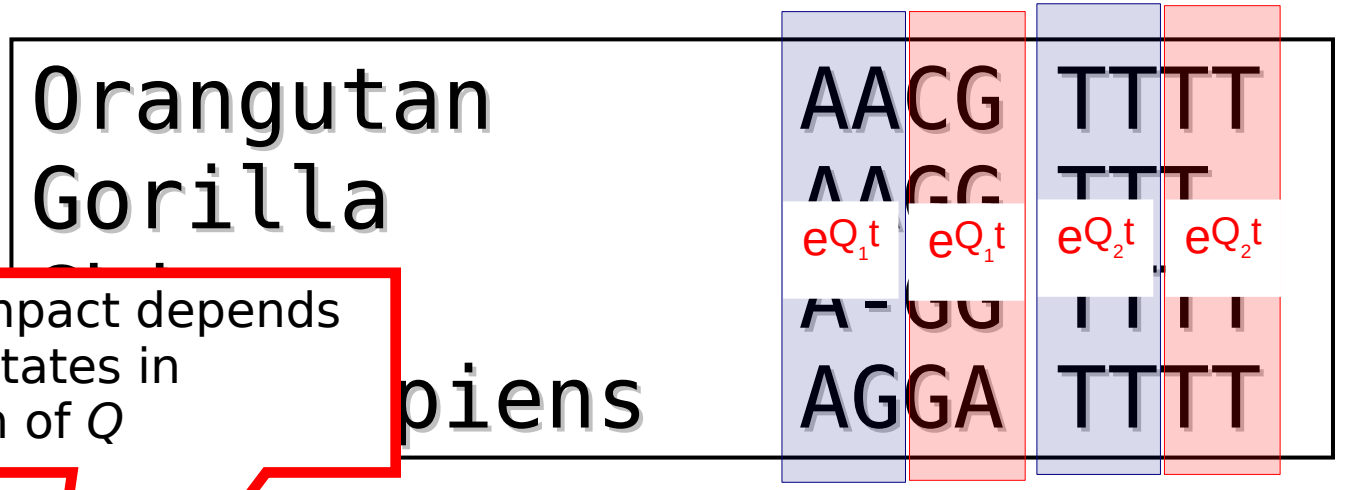
Works well when we have more processors than partitions:
However we will need to compute: $P(t)=e^{Qt}$ for each partition at each processor!

Data Distribution II



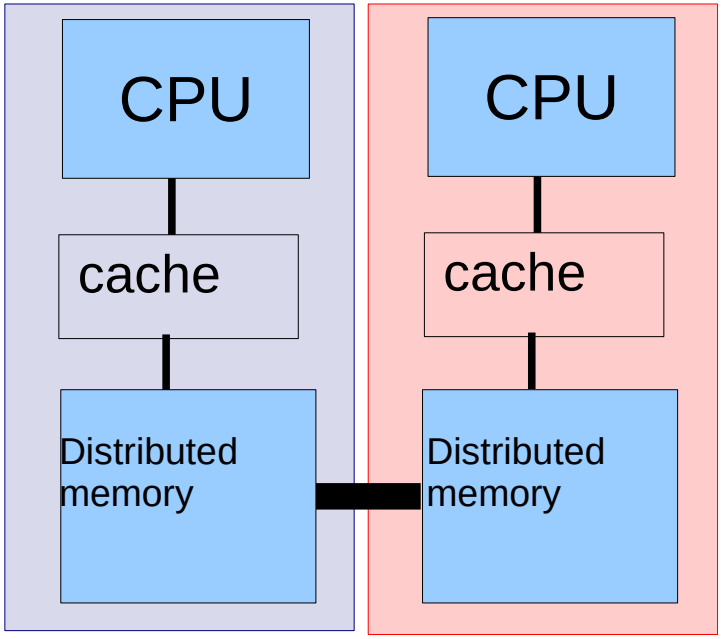
Works well when we have more processors than partitions:
However we will need to compute: $P(t)=e^{Qt}$ for each partition at each processor!

Data Distribution II

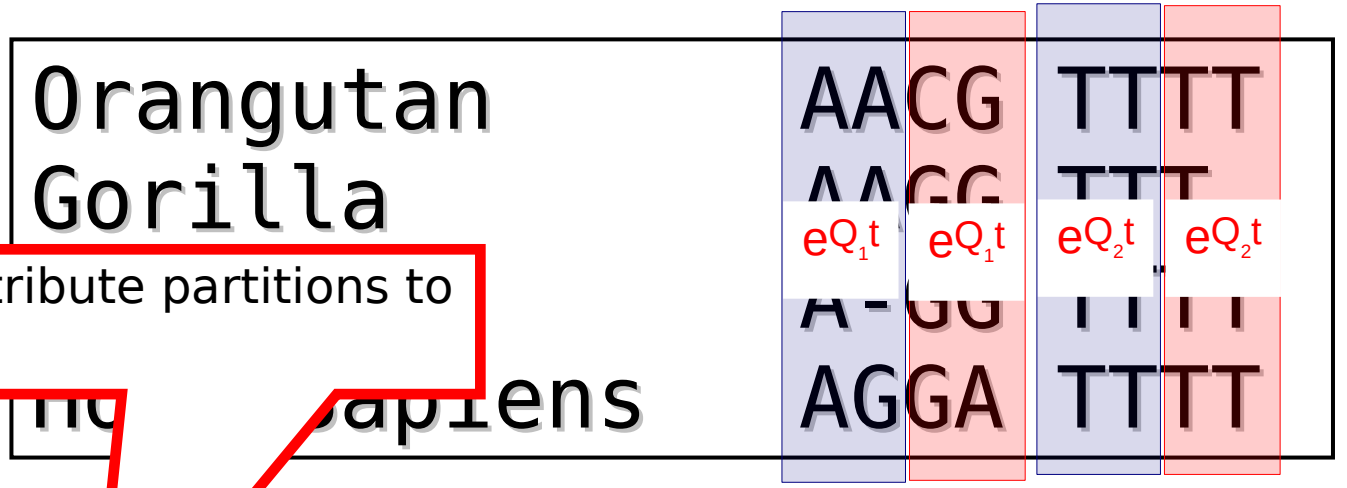


Performance impact depends on number of states in data/dimension of *Q*

Works well when we have more processors than partitions:
However we will need to compute: $P(t)=e^{Qt}$ for each partition at each processor!

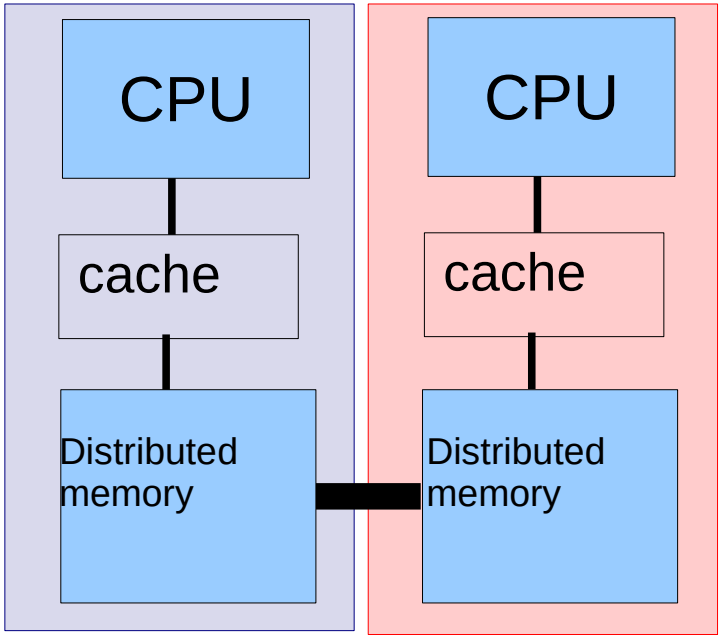


Data Distribution II

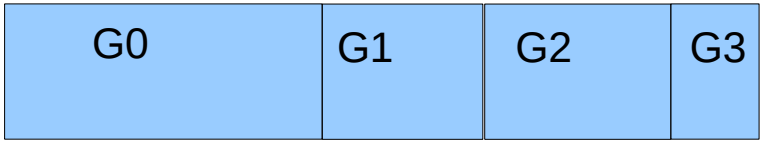


How do we distribute partitions to processors?

Works well when we have more processors than partitions:
However we will need to compute: $P(t)=e^{Qt}$ for each partition at each processor!

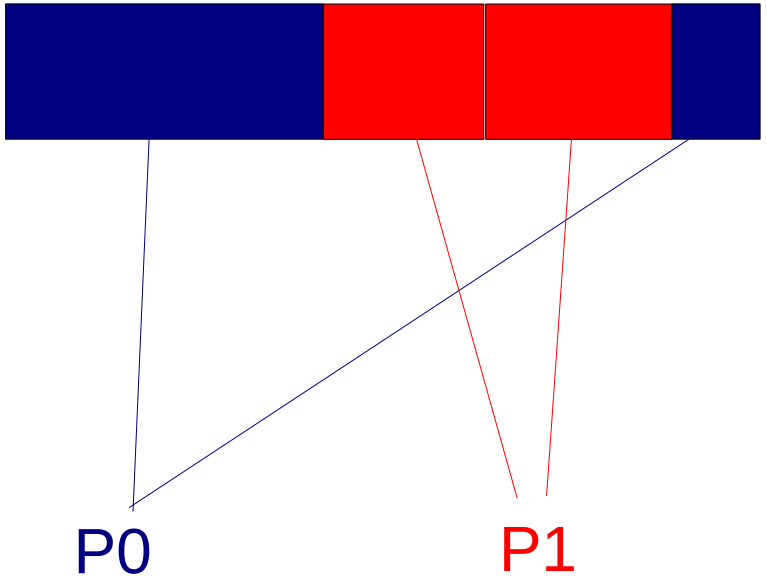


Load Balance I

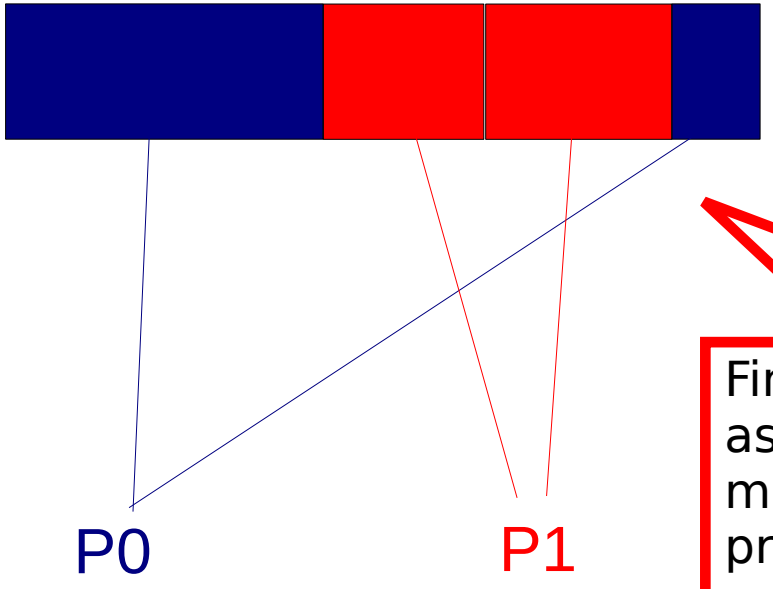


P0 P1

Load Balance I



Load Balance I



Find the partition-to-processor assignment such that the maximum number of sites per processor is minimized
→ this is NP-hard

Load Balance I

- The **multiprocessor job scheduling problem** in phylogenetics
  - Problem when #partitions >> #cores
  - Tested per-site (cyclic/modulo) data distribution versus per partition data distribution
  - We used the Longest Processing Time (LPT) heuristics for assigning partitions to processors
  - 25 taxa, 220,000 sites, 100 genes
    - GAMMA model

      naïve: **613** secs

      LPT: **550** secs
    - CAT model

      naïve: **298** secs

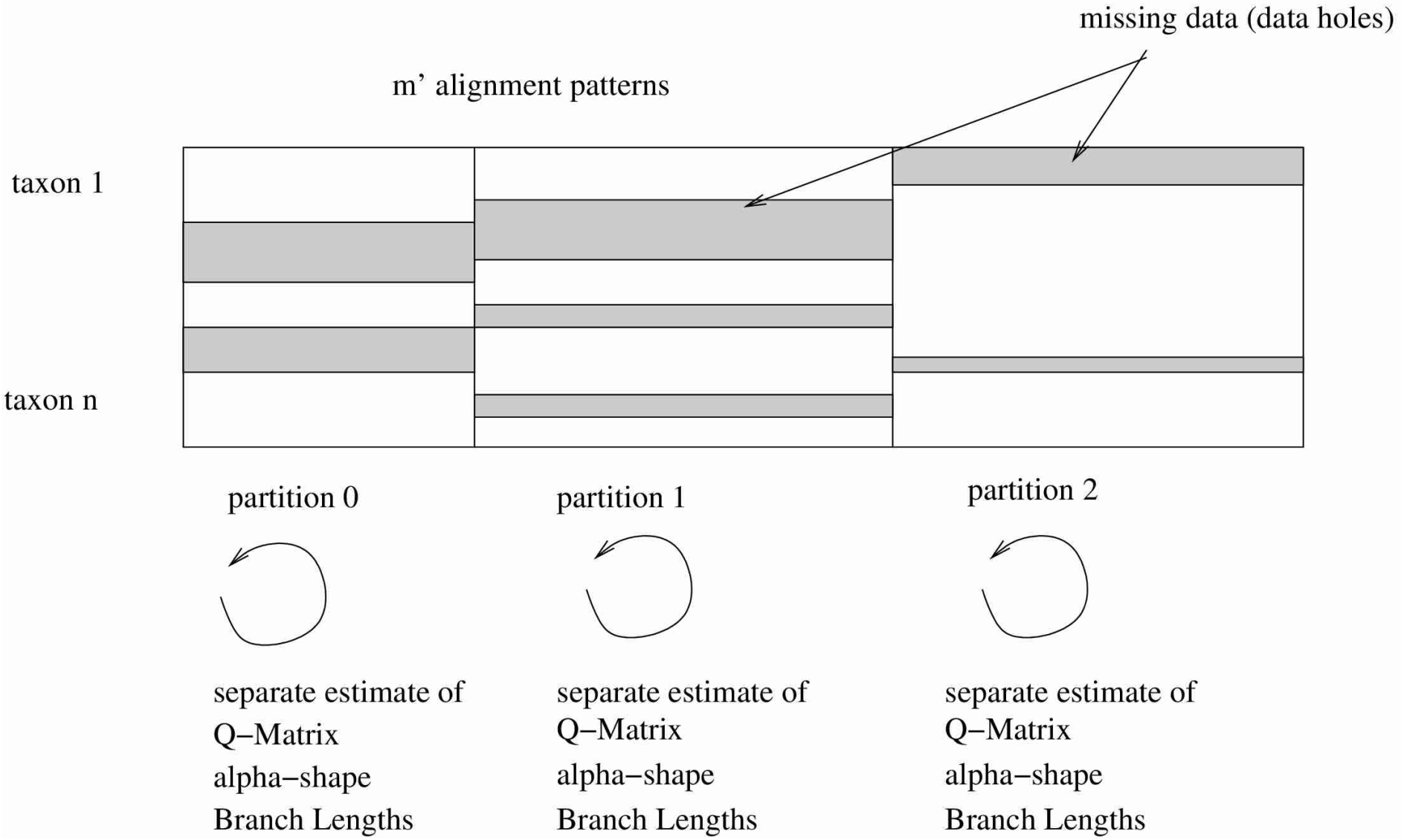
      LPT: **127** secs
  - Larger protein dataset under Γ model of rate heterogeneity: 10-fold performance improvement!

J. Zhang, A. Stamatakis: "The Multi-Processor Scheduling Problem in Phylogenetics", 11th IEEE HICOMB workshop (in conjunction with IPDPS 2012).

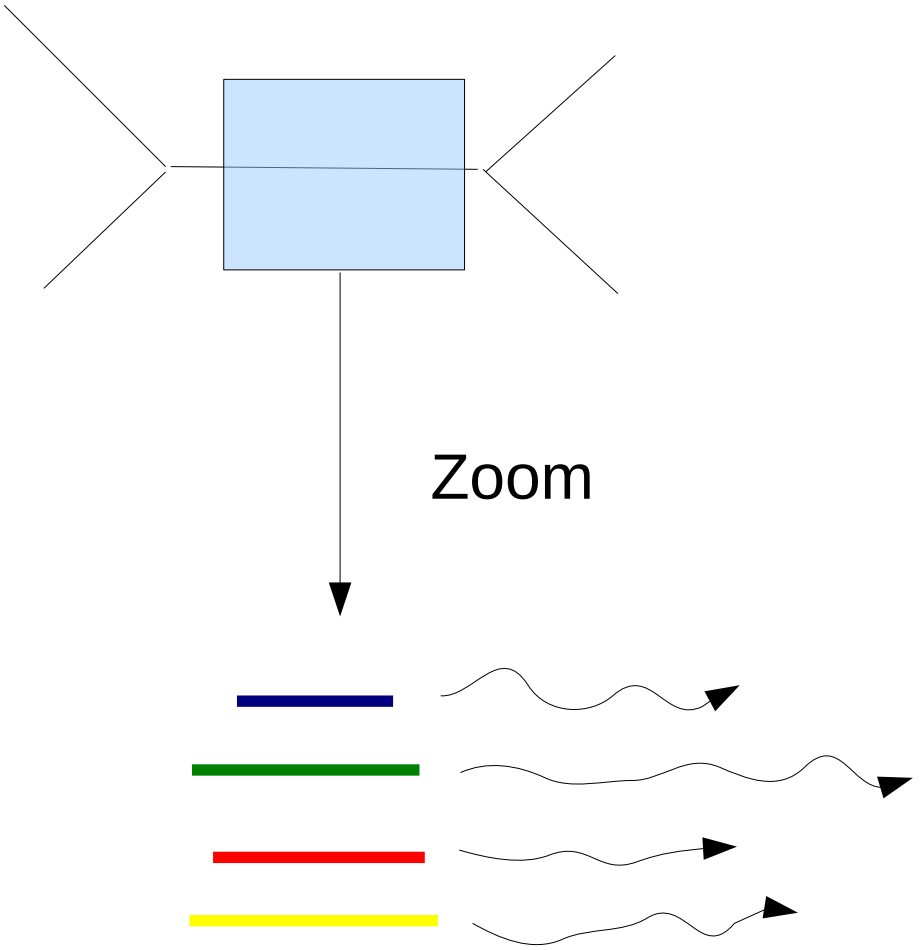
LPT heuristics for multi-processor scheduling

- Sort jobs (partitions) by processing length (partition length) in decreasing order
- Remove a job (partition) from the sorted list and assign it to the processor with the earliest end time (the smallest sum of partition lengths/number of sites)
- Repeat until the sorted list is empty
- Upper bound: *4/3 – 1/(3p) * OPT,* where *p* is the number of processors
- Graham, R. L.: "Bounds on Multiprocessing Timing Anomalies". *SIAM Journal on Applied Mathematics* 17 (2): 416–429, 1969.
- Remark: LPT works surprisingly well (see our paper on the phylogenetic problem where we also tested other heuristics)

Partitioned Branch Lengths & other parameters



Load-Balance II



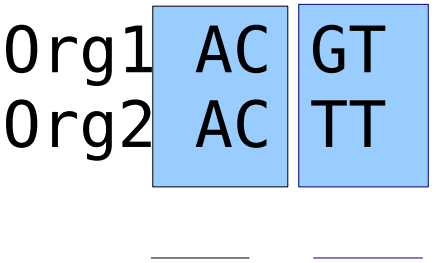
Synchronization Points

- Assume 10 branches
- Each branch requires 10 Newton-Raphson Iterations
- Each NR Iteration requires a synchronization via a reduction operation
- One branch/partition at a time: 100 sync. points, less work (only one partition) per sync. point
- All branches concurrently: 10 sync. points, more work per sync. point
- Branches will need distinct number of operations
- Add convergence state → bit vector

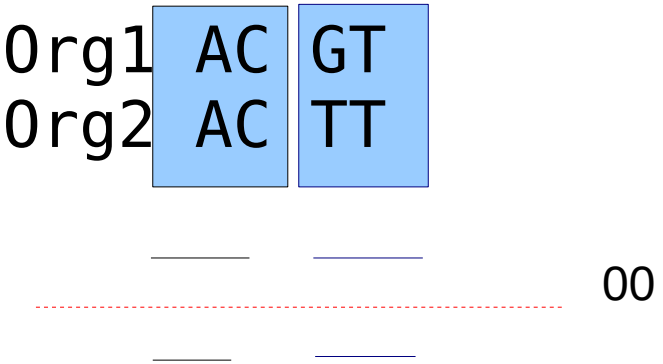
Synchronization Points

```
Org1 AC GT
Org2 AC TT
```

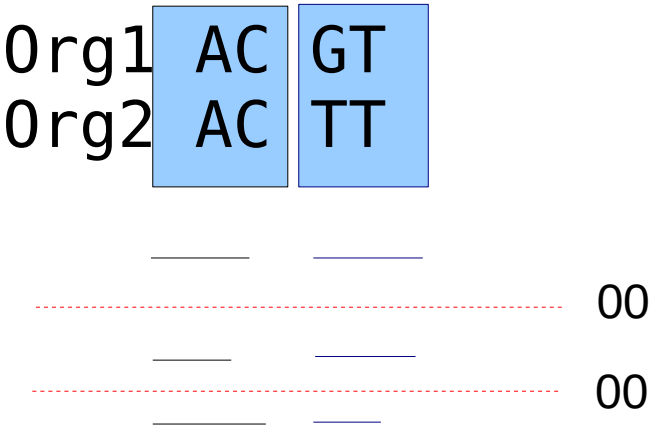
Synchronization Points



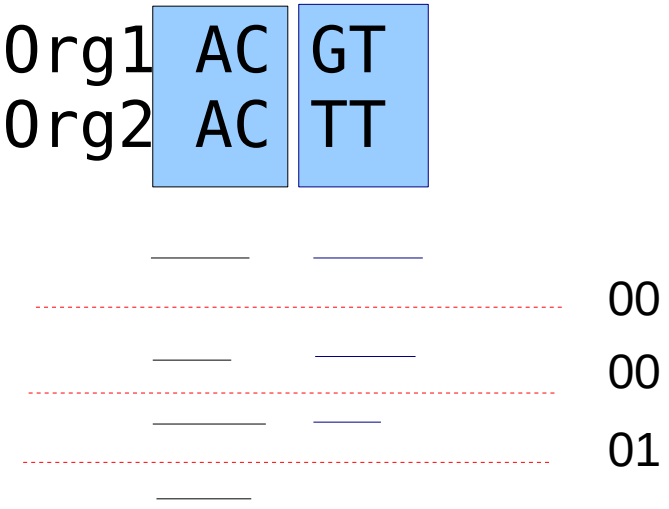
Synchronization Points



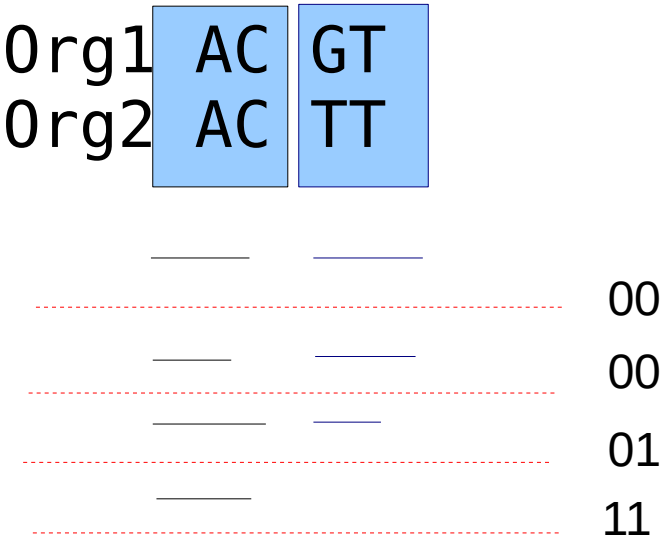
Synchronization Points



Synchronization Points

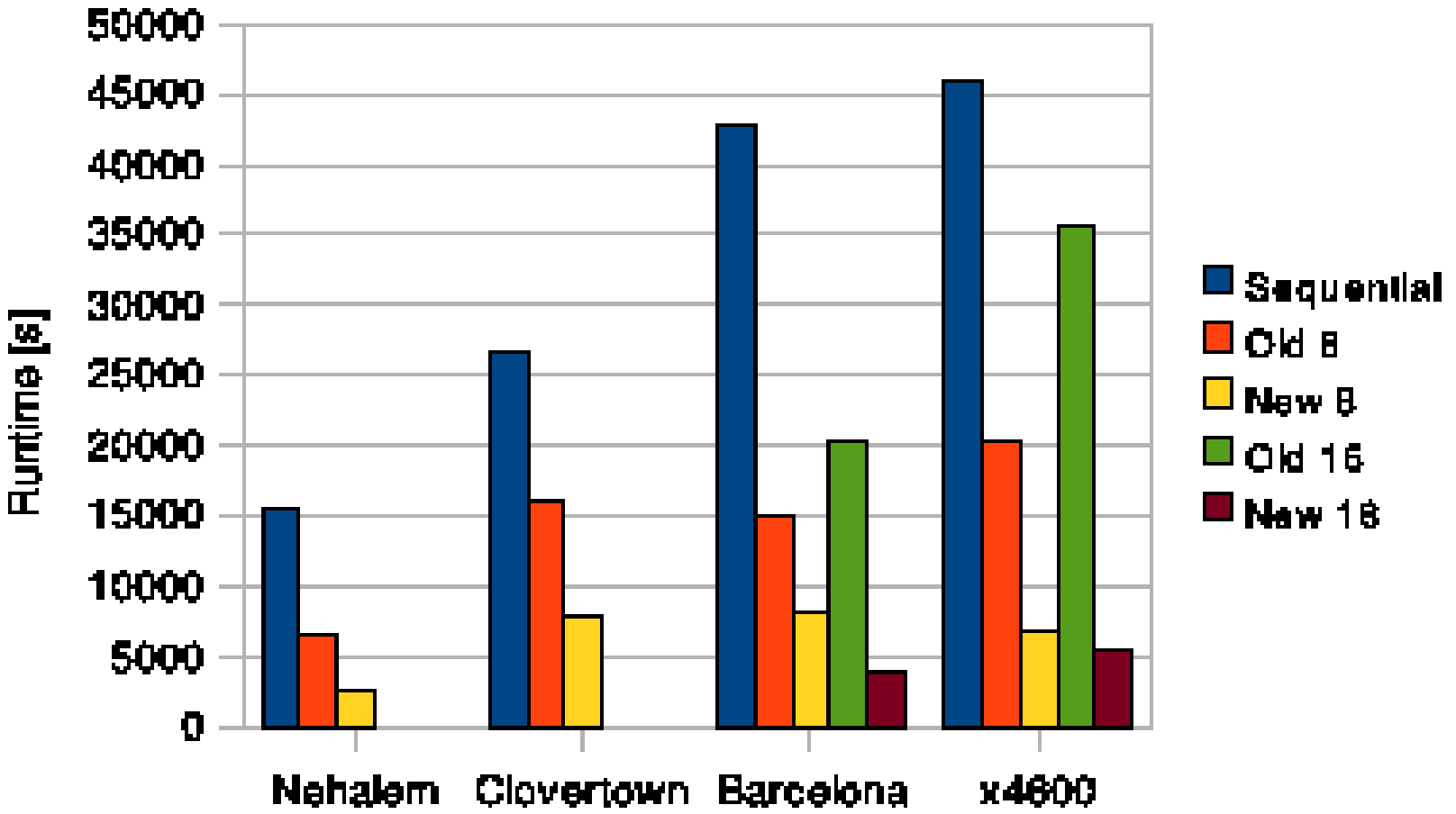


Synchronization Points



In this example: 4 instead of 7 sync points!

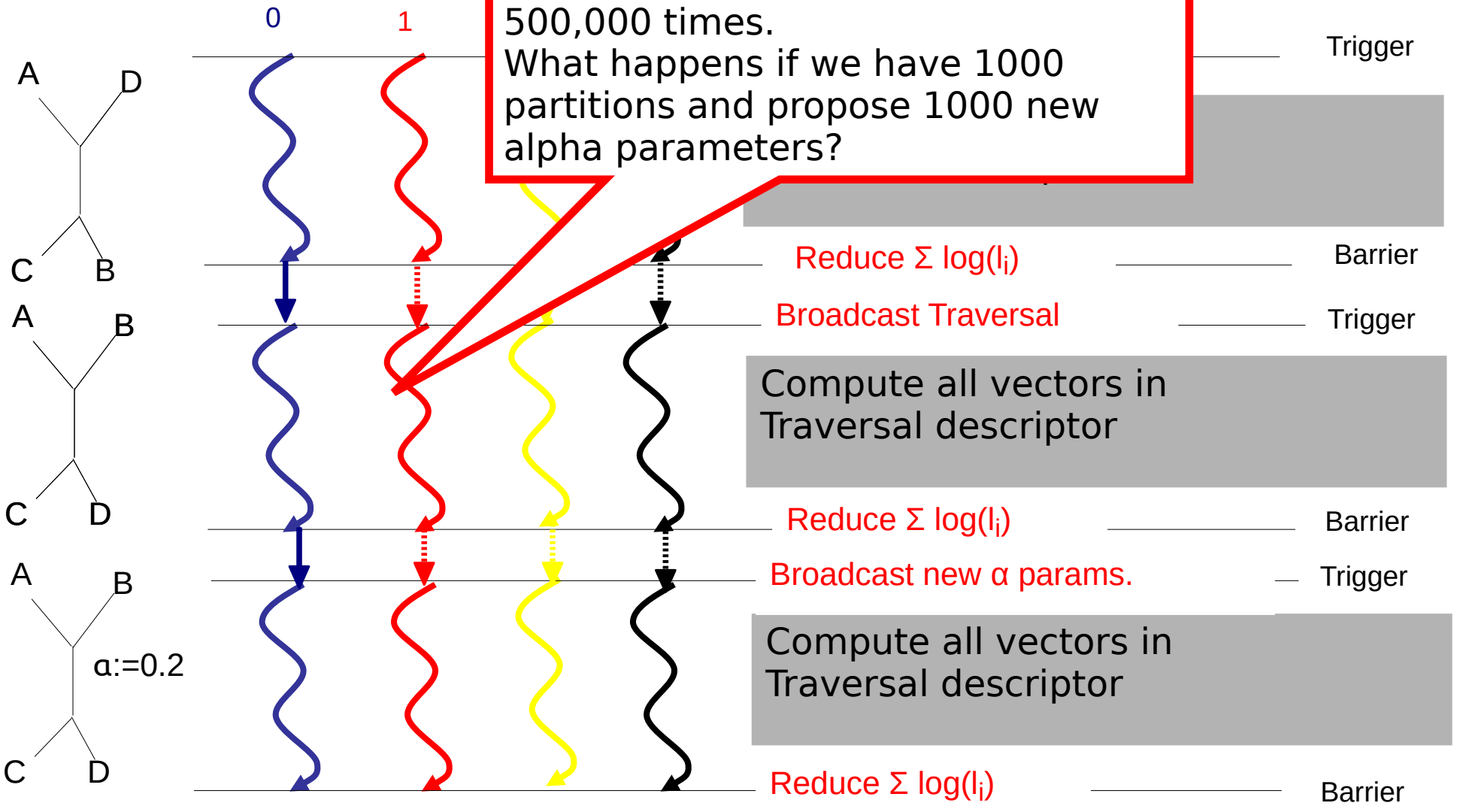
Load Balance II



A. Stamatakis, M. Ott: "Load Balance in the Phylogenetic Likelihood Kernel". Proceedings of ICPP 2009, Vienna, Austria, September 2009.

Classic Fork-Join with

For good parallel performance: the broadcast must be fast!
Remember: 10 secs 16 cores approx 500,000 times.
What happens if we have 1000 partitions and propose 1000 new alpha parameters?



Alternative MPI parallelization