

# **A parallel algorithm for the extraction of structured motifs**

Alexandra Carvalho

MEIC 2002/03

# Plan of the talk

- Biological model of regulation
  - Nucleic acids: DNA and RNA
  - Classification of living organisms: Prokaryotes and Eukaryotes
  - Transcription and Translation
  - Promoter and Regulatory Sequences
- Computational model of regulation
  - Suffix tree and generalized suffix tree
  - Single models extraction [**M.-F. Sagot, *Latin*, 1998**]
  - Structured models extraction [**L. Marsan and M.-F. Sagot, *J. Computational Biology*, 2000**]
  - Parallelization [**A. Carvalho, A. Freitas, A. Oliveira and M.-F. Sagot, submitted, 2003**]
    - The PARTITION UP TO  $\varepsilon$  problem
      - The SimpleCut algorithm
    - The tree partition problem
      - The PSMILE algorithm
    - Experimental results

# Nucleic acids

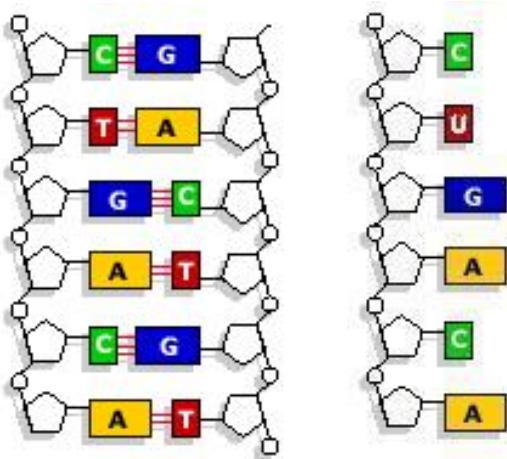
Nucleotides:

- storage and retrieval of biological information
- building blocks for the construction of nucleic acids

# Nucleic acids

Nucleotides:

- storage and retrieval of biological information
- building blocks for the construction of nucleic acids



Two main types of nucleic acids:

- DNA - DeoxyriboNucleic Acid:
  - contain the bases A, C, G, and T
  - double-stranded molecule
- RNA - RiboNucleic Acid:
  - contain the bases A, C, G, and U
  - single-stranded molecule

# Classification of living organisms

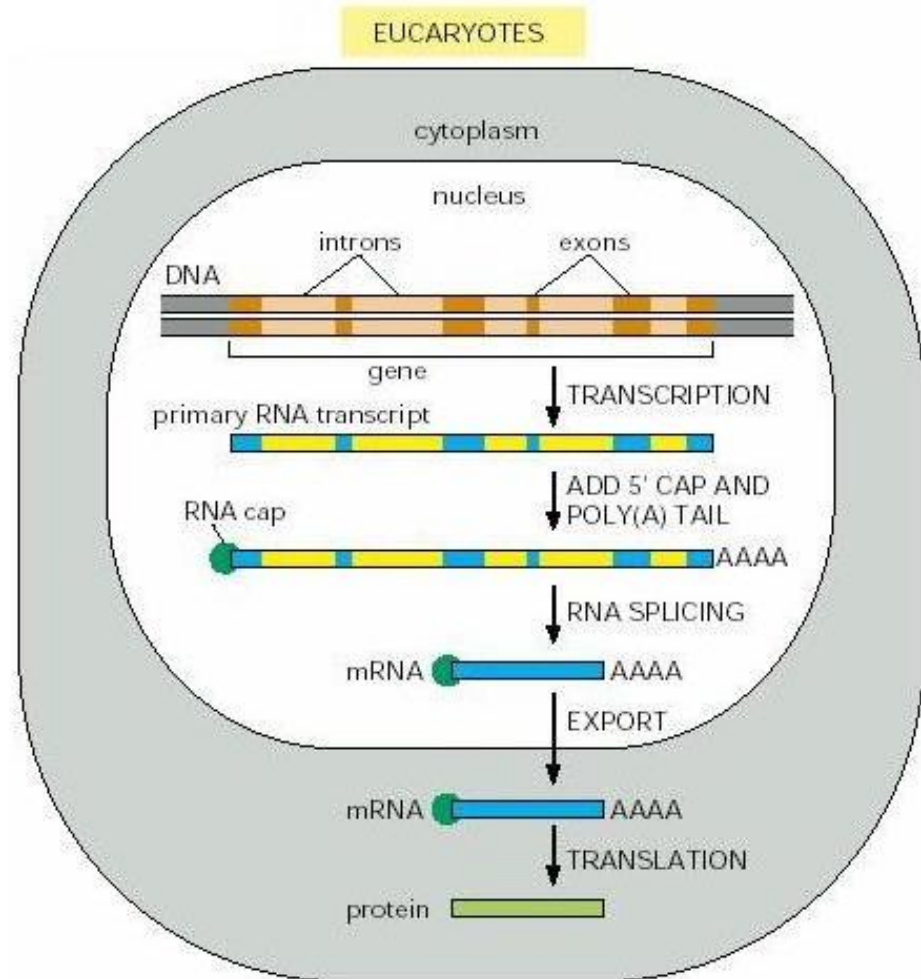
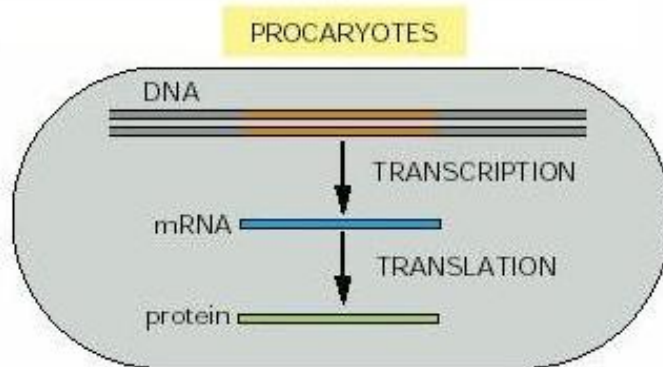
## Prokaryotes:

- Greek words: *pro*  $\equiv$  "before" and *karyon*  $\equiv$  "nucleus"
- bacteria and prokaryotes are generally used interchangeably
- most prokaryotes live as single-celled organisms

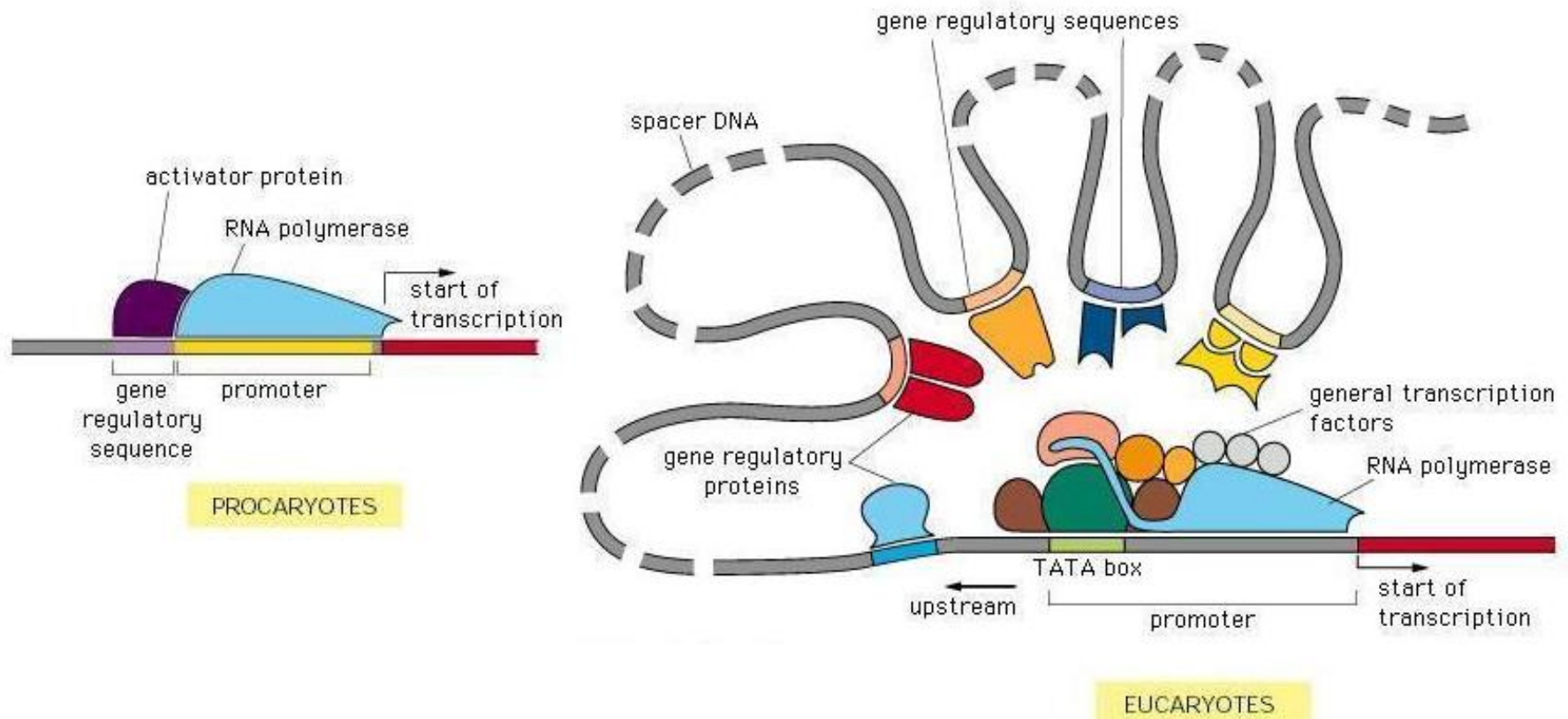
## Eukaryotes:

- Greek words: *eu*  $\equiv$  "well" and *karyon*  $\equiv$  "nucleus"
- *yeast* is an eukaryotic single-celled organism

# Transcription and Translation



# Promoter and Regulatory Sequences



# Structured motifs

**Definition.** *model*

A model is an element in  $\Sigma^+$ .

**Definition.** *structured model*

A structured model is a pair  $(m, d)$  where:

- $m = (m_i)_{1 \leq i \leq p}$ , denoting the  $p$  boxes
- $d = (d_{\min_i}, d_{\max_i}, \delta_i)_{1 \leq i \leq p-1}$ , denoting the  $p - 1$  intervals of distance



# Structured motifs

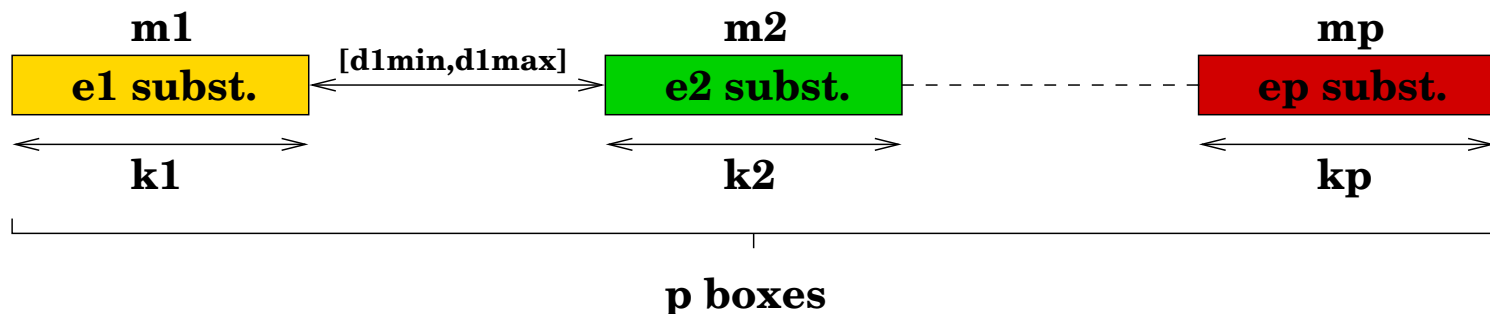
## Definition. *model*

A model is an element in  $\Sigma^+$ .

## Definition. *structured model*

A structured model is a pair  $(m, d)$  where:

- $m = (m_i)_{1 \leq i \leq p}$ , denoting the  $p$  boxes
- $d = (d_{\min_i}, d_{\max_i}, \delta_i)_{1 \leq i \leq p-1}$ , denoting the  $p - 1$  intervals of distance



An attempt to model the combinatorics of regulation

# Structured motifs

**Definition. *e*-occurrence**

A model  $m$  *e*-occurs in the input sequences if exists  $u$  in the input sequences such that  $\text{HammingDistance}(m, u) \leq e$  (minimum number of substitutions to transform  $u$  into  $m$ ).

# Structured motifs

**Definition. *e-occurrence***

A model  $m$   $e$ -occurs in the input sequences if exists  $u$  in the input sequences such that  $\text{HammingDistance}(m, u) \leq e$  (minimum number of substitutions to transform  $u$  into  $m$ ).

**Definition. *valid model, quorum***

A model is valid if  $e$ -occurs in at least  $q$  input sequences, where  $q$  is called the quorum.

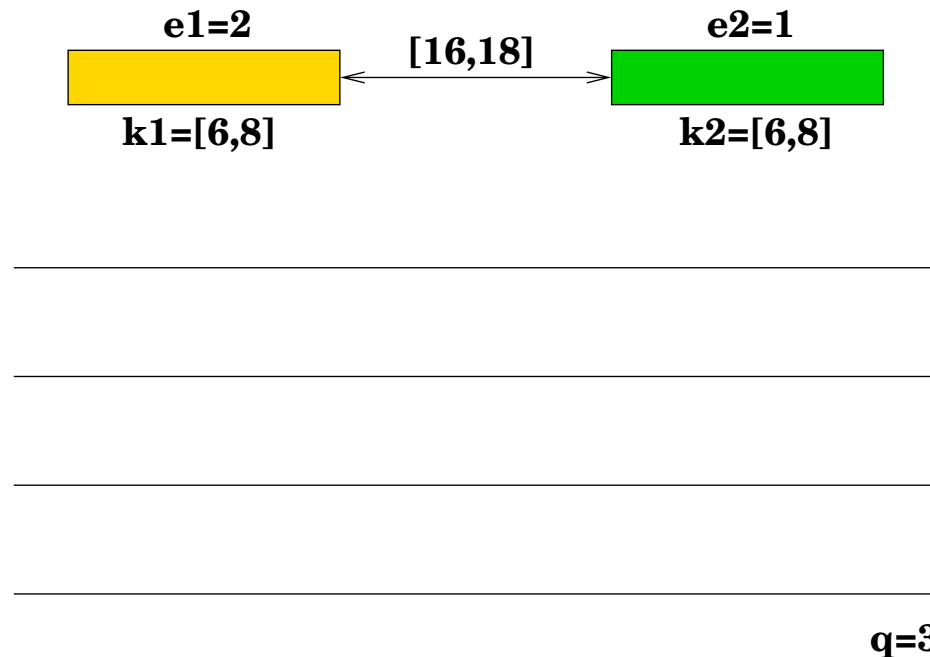
# Structured motifs

**Definition. *e-occurrence***

A model  $m$   $e$ -occurs in the input sequences if exists  $u$  in the input sequences such that  $\text{HammingDistance}(m, u) \leq e$  (minimum number of substitutions to transform  $u$  into  $m$ ).

**Definition. *valid model, quorum***

A model is valid if  $e$ -occurs in at least  $q$  input sequences, where  $q$  is called the quorum.



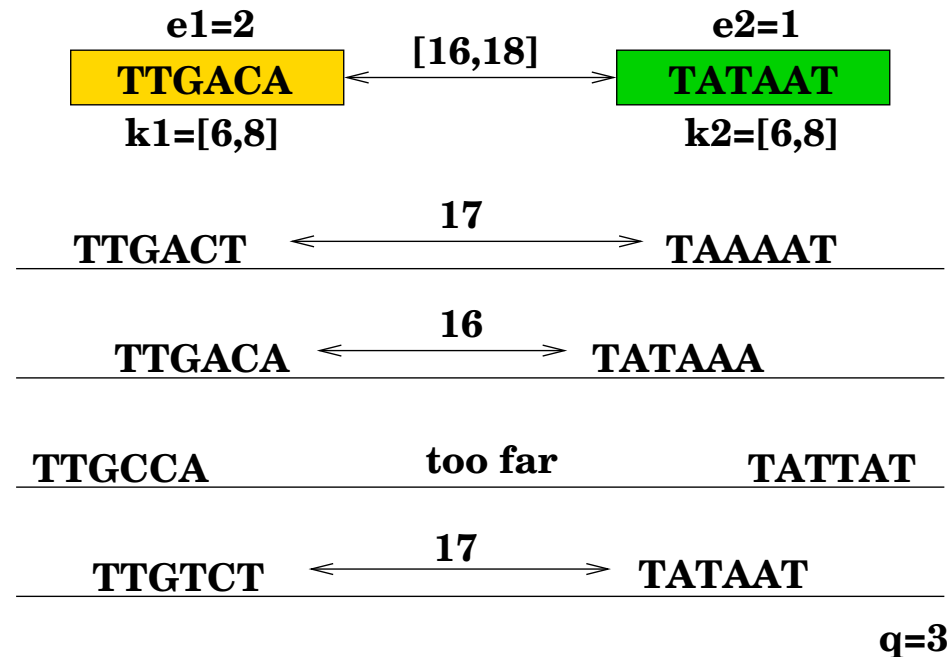
# Structured motifs

## Definition. *e-occurrence*

A model  $m$   $e$ -occurs in the input sequences if exists  $u$  in the input sequences such that  $\text{HammingDistance}(m, u) \leq e$  (minimum number of substitutions to transform  $u$  into  $m$ ).

## Definition. *valid model, quorum*

A model is valid if  $e$ -occurs in at least  $q$  input sequences, where  $q$  is called the quorum.



# Input sequences

>strand + guaB inositol-monophosphate dehydrogenas

CTTTCCGTTATCTAAATATTTCAACTCTTTCCCGCTTCCTTGACATGCTCTTGGCTAGTTGATAATCT  
ACATATAATATTTTGCCGAAAA

>strand - yaaC yaaC

TTTTCGGCAAAATATTATATGTAGATTATCAACTAGCCAAGAGCATGTCAAGGAAGCGGGAAAGAGTT  
GAAATATTTAGATAACGGAAAG

>strand + yaaJ similar to hypothetical proteins

CCGTTTCAGTTATAGTTAATATGTAGCCTTTTTTAGGCAATGAAAAAACTTTGAAA

>strand - yaaI similar to isochorismatase

TTTCAAAGTTTTTTTCATTGCCTAAAAAGGCTACATATTA ACTATAACTGAAACGG

>strand + metS methionyl-tRNA synthetase

ATTTTATAAATATTTAATAAAGCTATTATCCTACTAAAAATCCTTTTAAATCAAGACTTTTCGAACCAA  
AGTTTTTTTATTTTCATTTGATTATATACGACAAAATTCGACACGAACAGACTTTTTTTTATTTTCATTAA  
AGATTTTTTAATTTTAATTATTCTTTTTTCAGGGCGTATGTATATATTCTTGATCTTAAAGGCTAAGATG  
GTATCATAGATAAAGGATAAATATAAATAATATTCATATATGATTTGCACTTATCGCCGCTCTCGTCC  
TTTGGGCGGGAGCTTTTTTGACATTCTGA

# Suffix Tree

**Definition.** *Suffix tree*

A suffix tree of a  $n$ -character string  $S$  is a rooted directed tree with exactly  $n$  leaves:

- leaves are numbered 1 to  $n$
- each internal node has at least two children
- each edge is labeled with a nonempty substring of  $S$
- no two edges out of a node can have edge-labels beginning with the same character

The key feature of the suffix tree is that for any leaf  $i$ , the label of the path from the root to the leaf  $i$  exactly spells out the suffix of  $S$  that starts at position  $i$ .

**Weiner, *IEEE Symposium on Switching and Automata Theory*, 1973**

**Ukkonen, *Algorithmica*, 1995**

**Theorem.** Suffix trees can be built in linear-time.

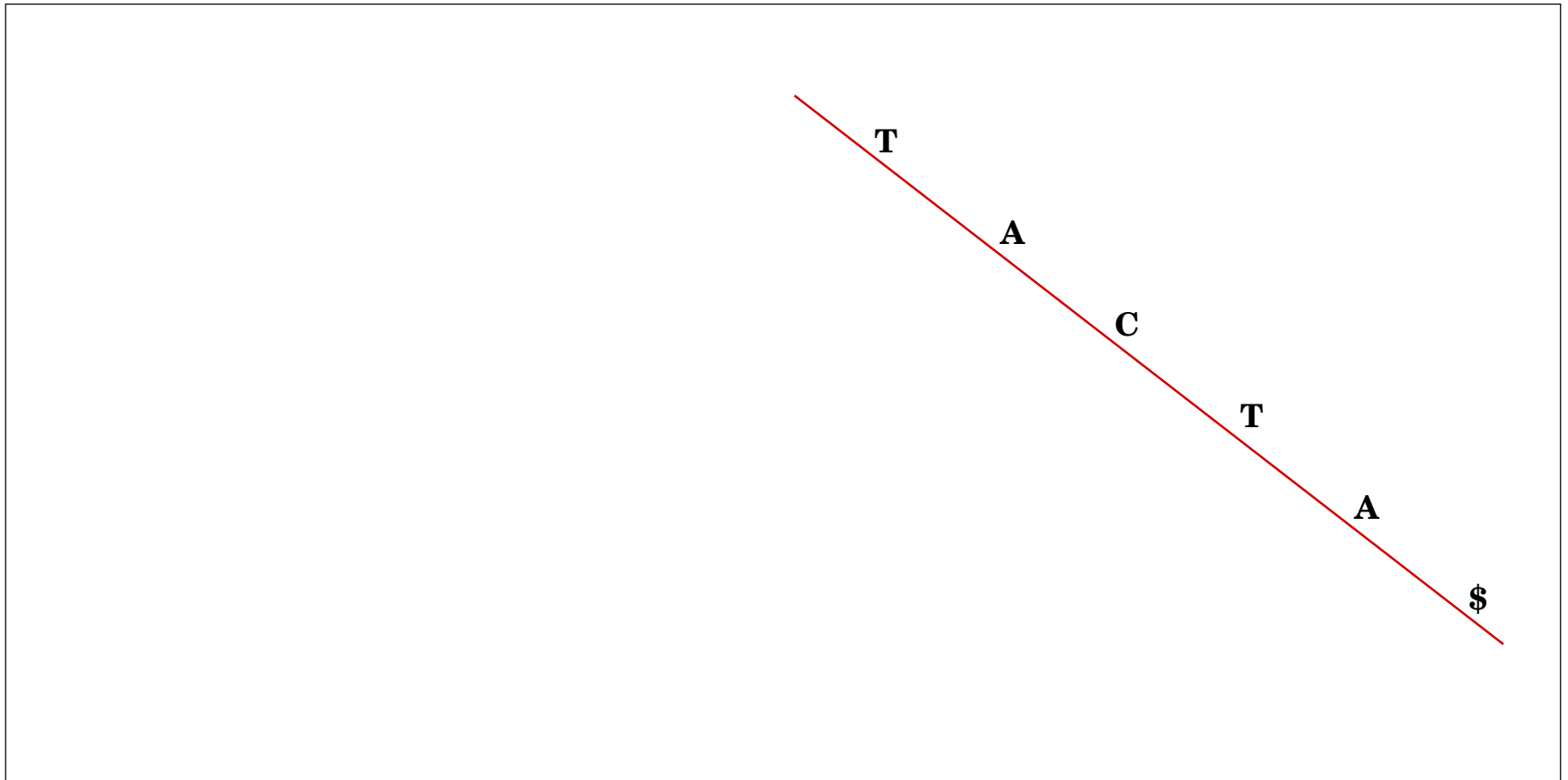
# Suffix Tree

Suffix tree for the string TACTA\$



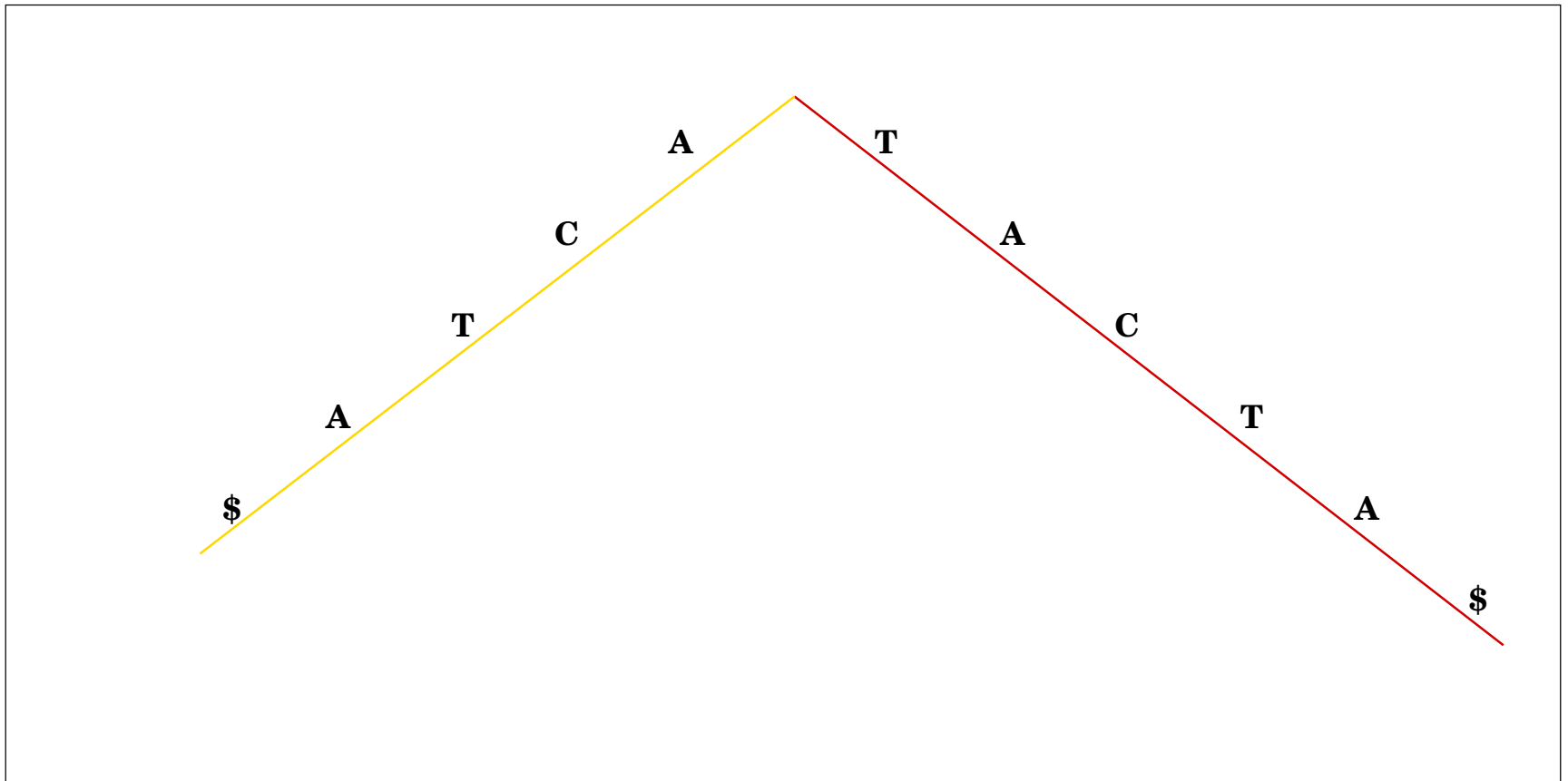
# Suffix Tree

Suffix tree for the string **TACTA\$**



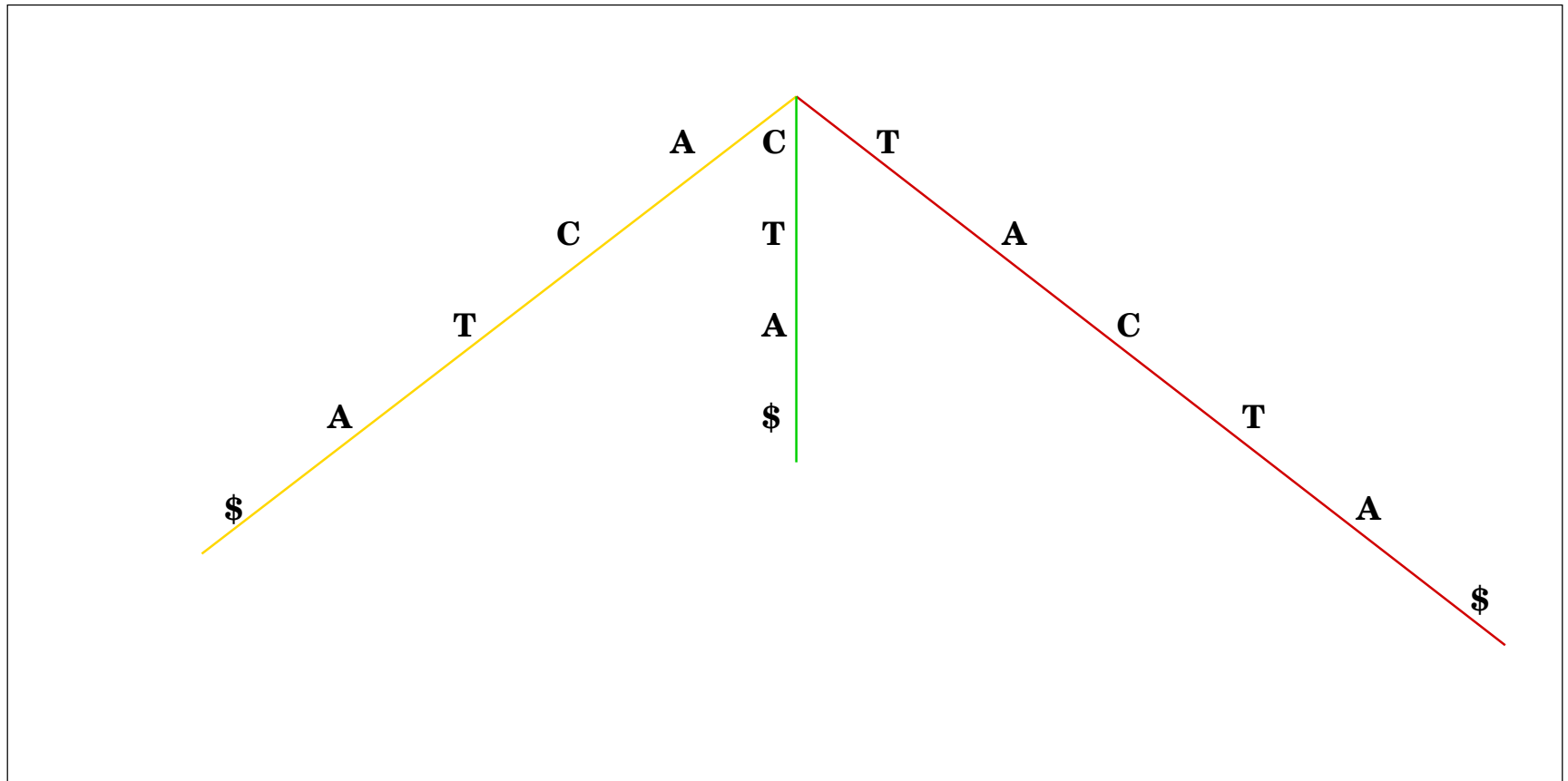
# Suffix Tree

Suffix tree for the string TACTA\$



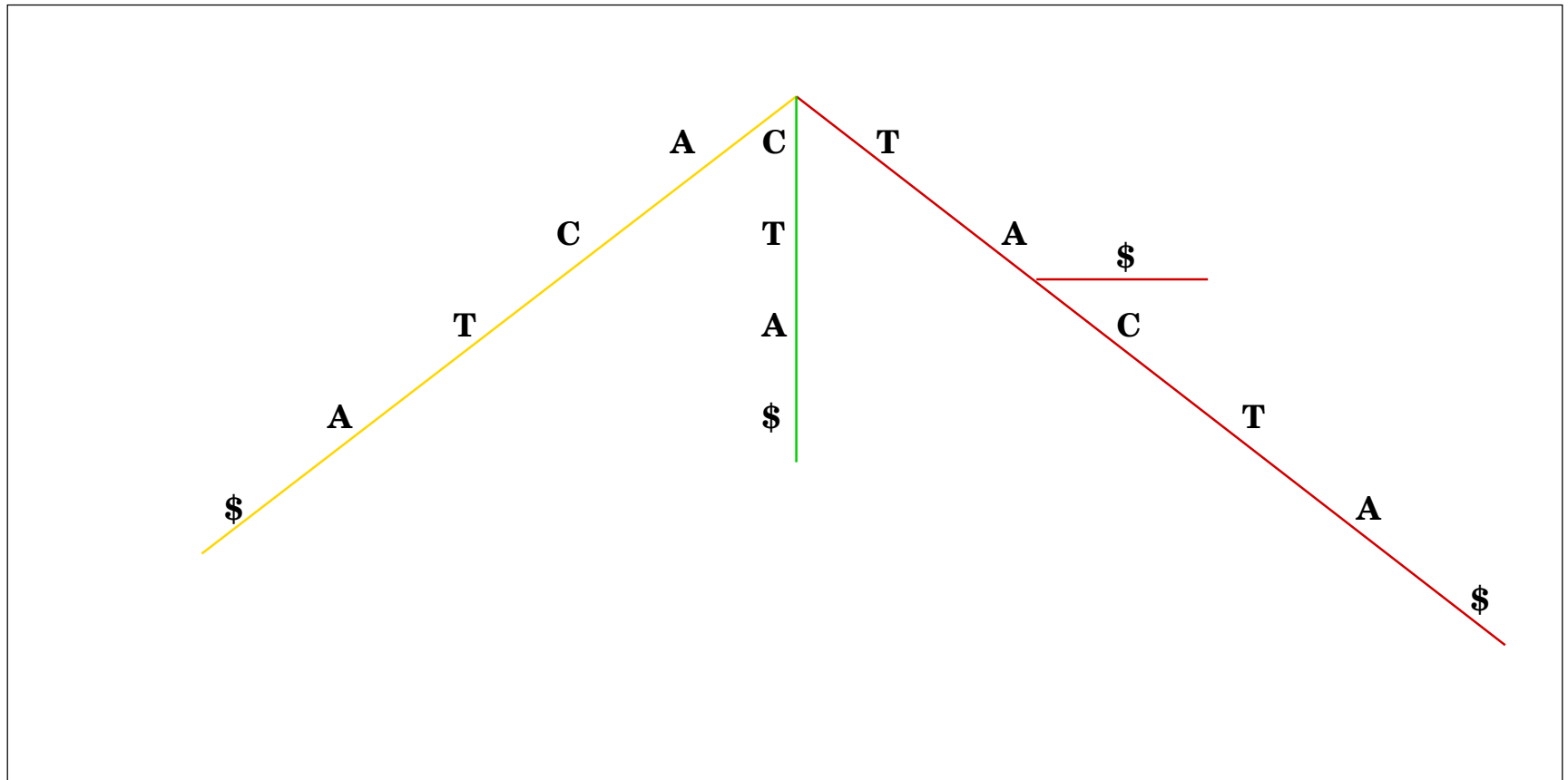
# Suffix Tree

Suffix tree for the string TACTA\$



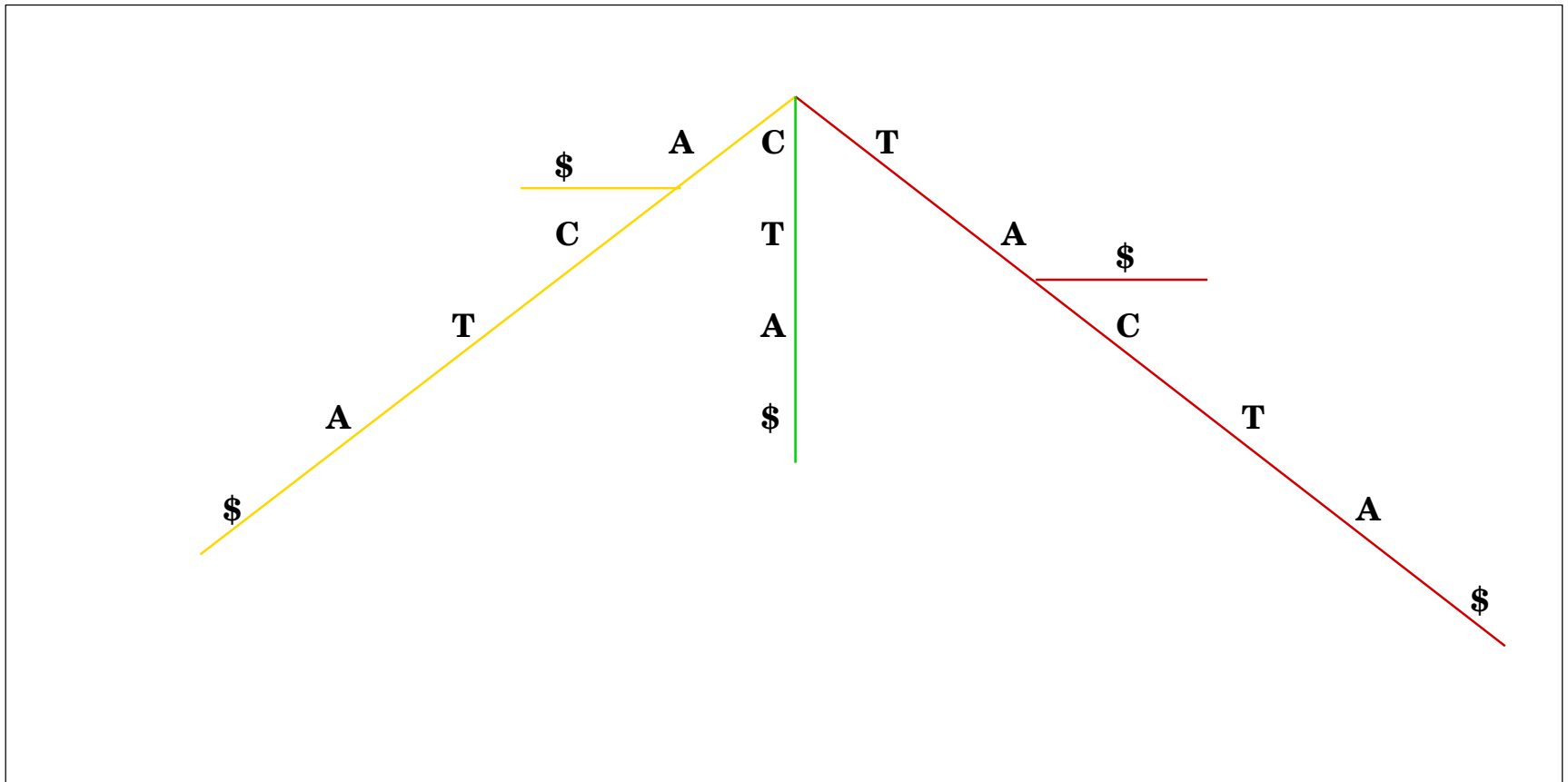
# Suffix Tree

Suffix tree for the string TACTA\$



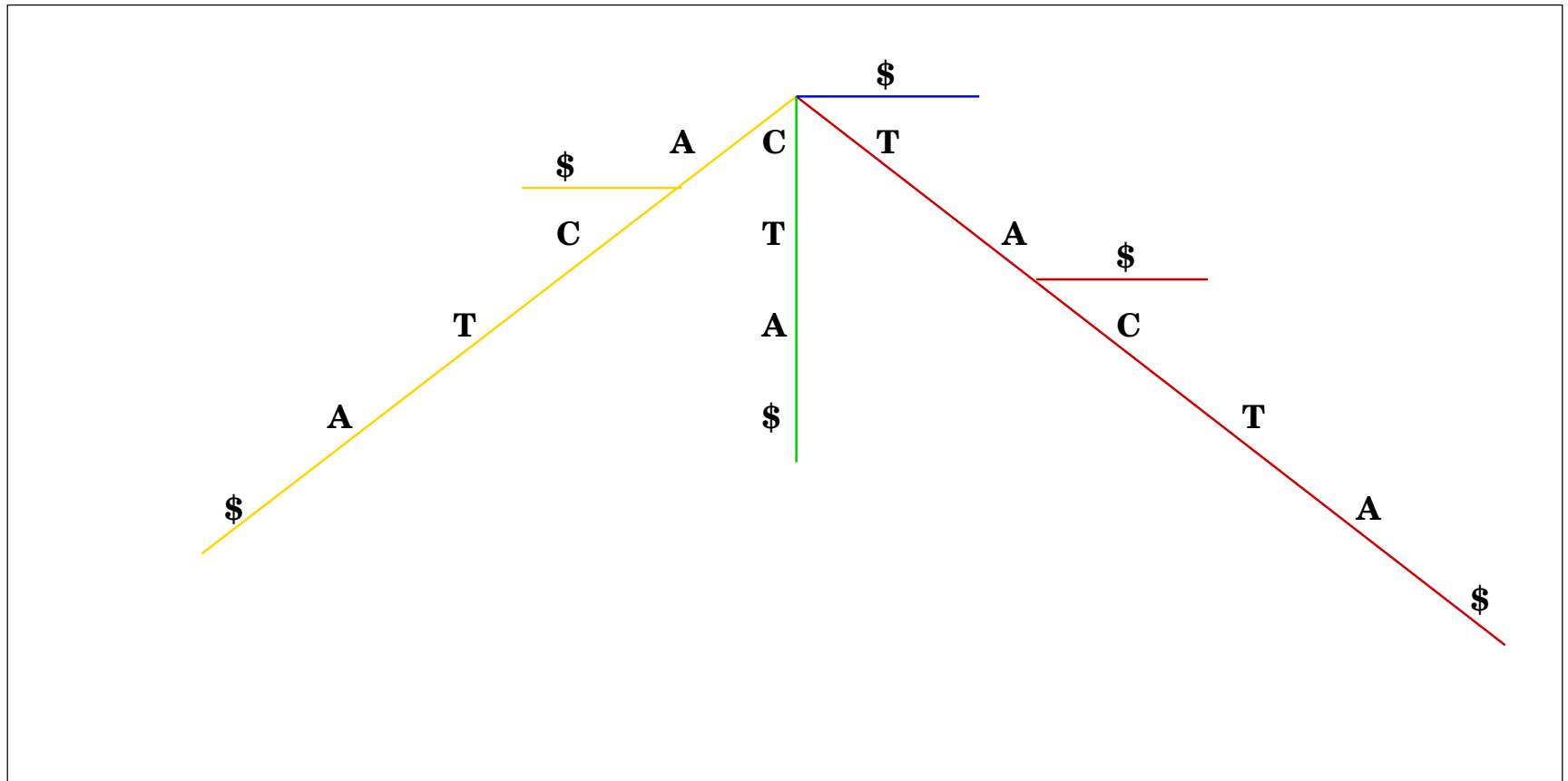
# Suffix Tree

Suffix tree for the string TACTA\$



# Suffix Tree

Suffix tree for the string TACTA\$

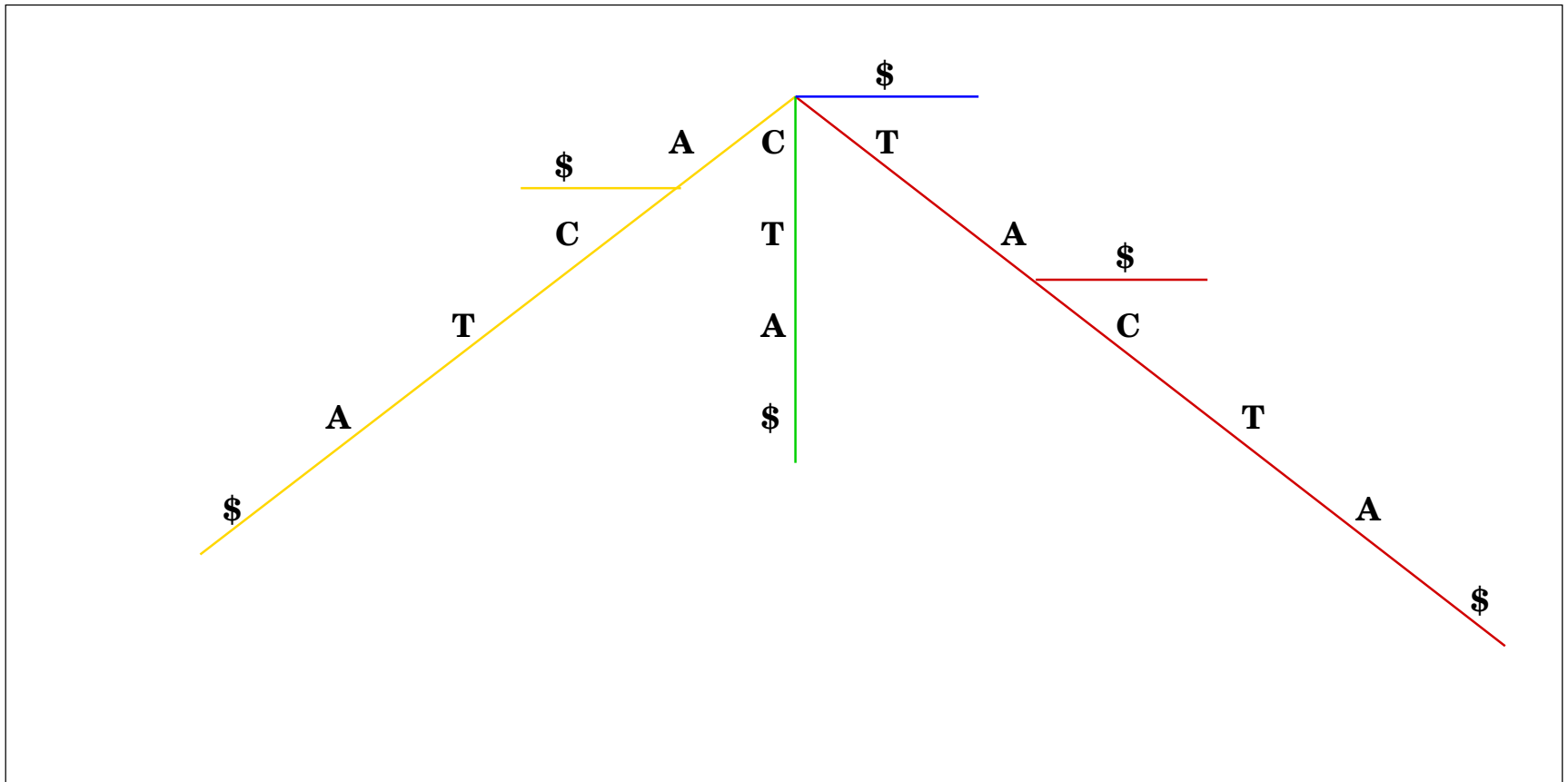


# Generalized Suffix Tree

Suffix tree for the strings TACTA\$ and CACTCA#

# Generalized Suffix Tree

Suffix tree for the strings TACTA\$ and CACTCA#



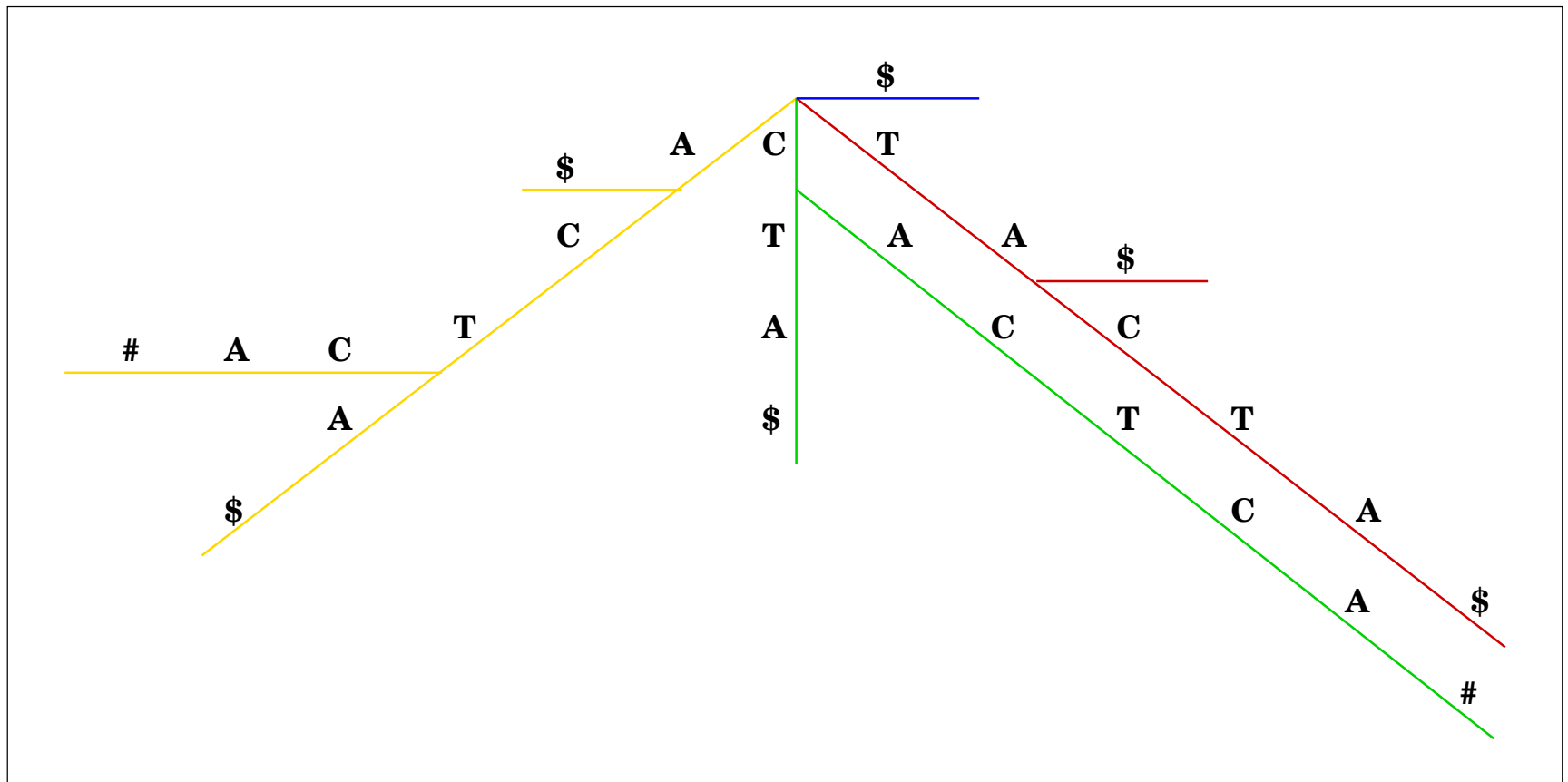


---



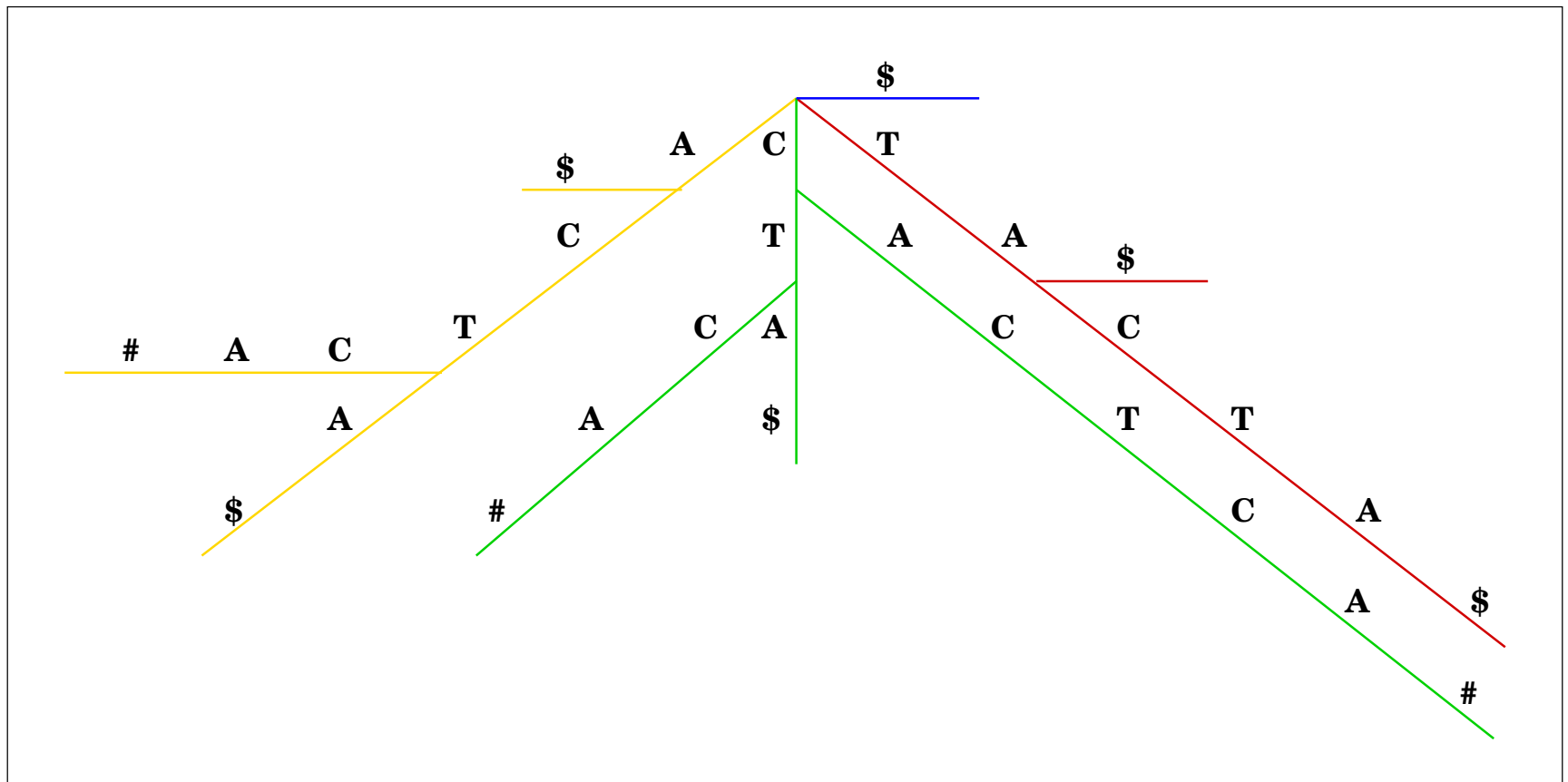
# Generalized Suffix Tree

Suffix tree for the strings TACTA\$ and CACTCA#



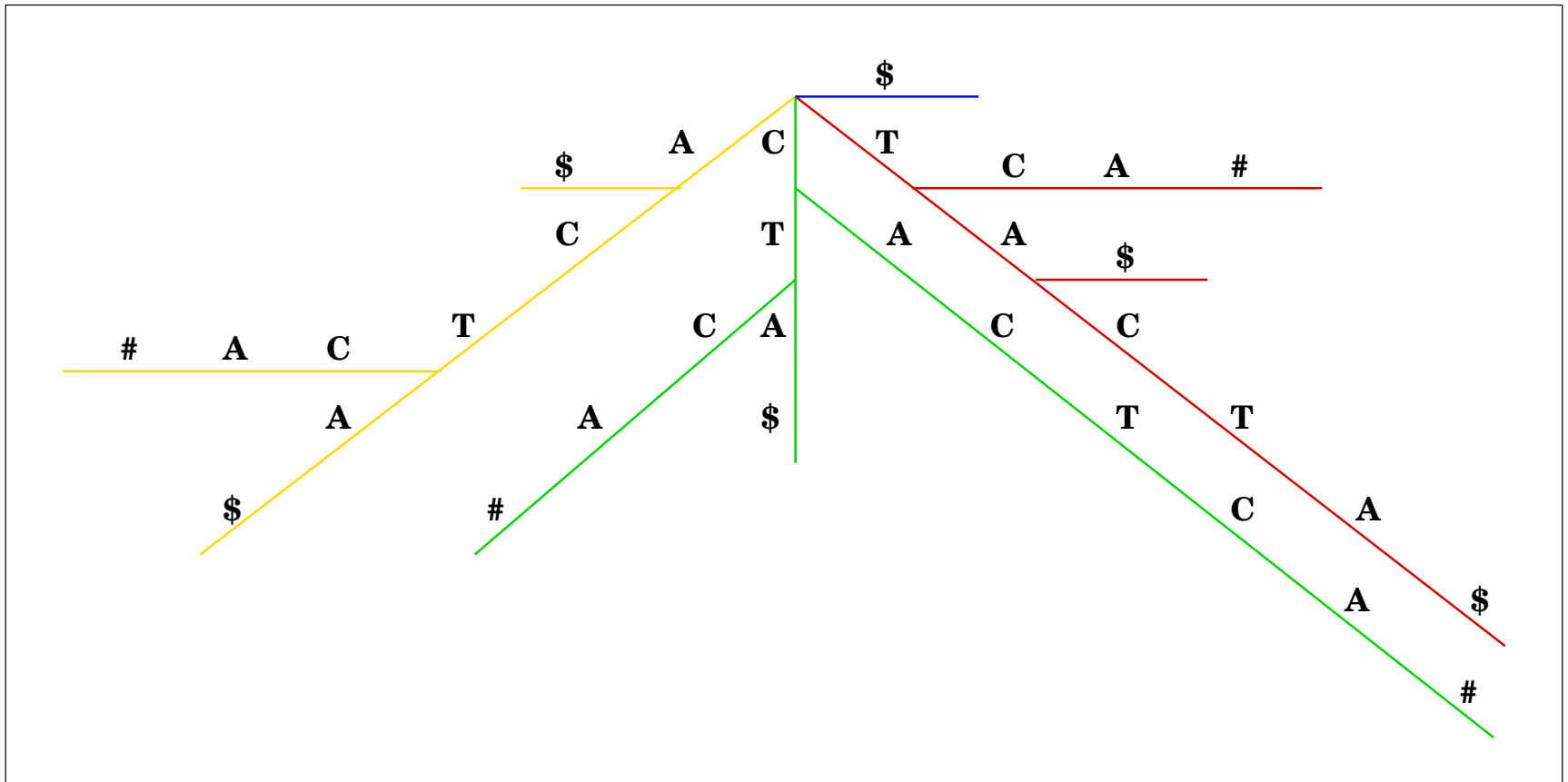
# Generalized Suffix Tree

Suffix tree for the strings TACTA\$ and CACTCA#



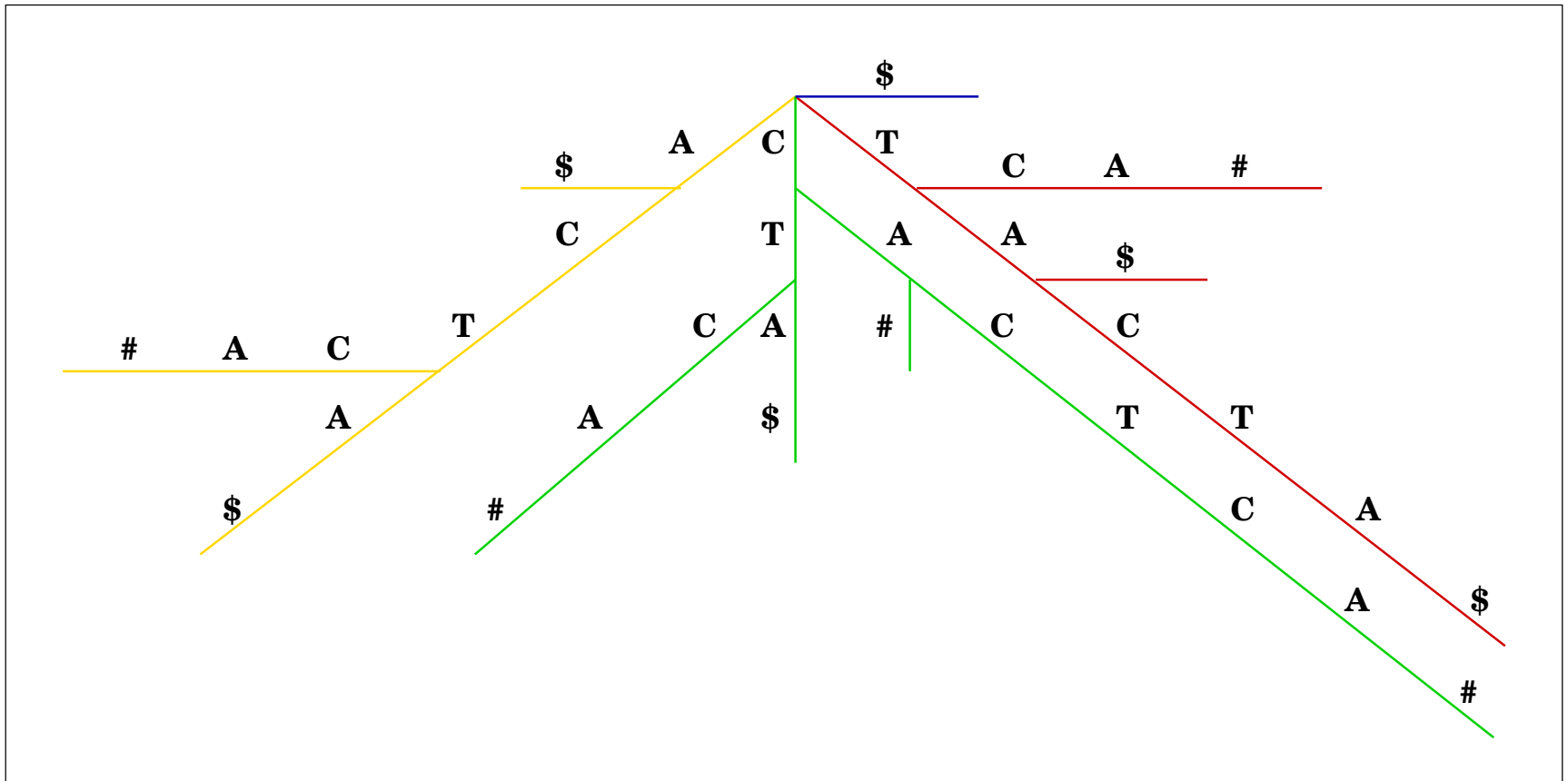
# Generalized Suffix Tree

Suffix tree for the strings TACTA\$ and CACTCA#



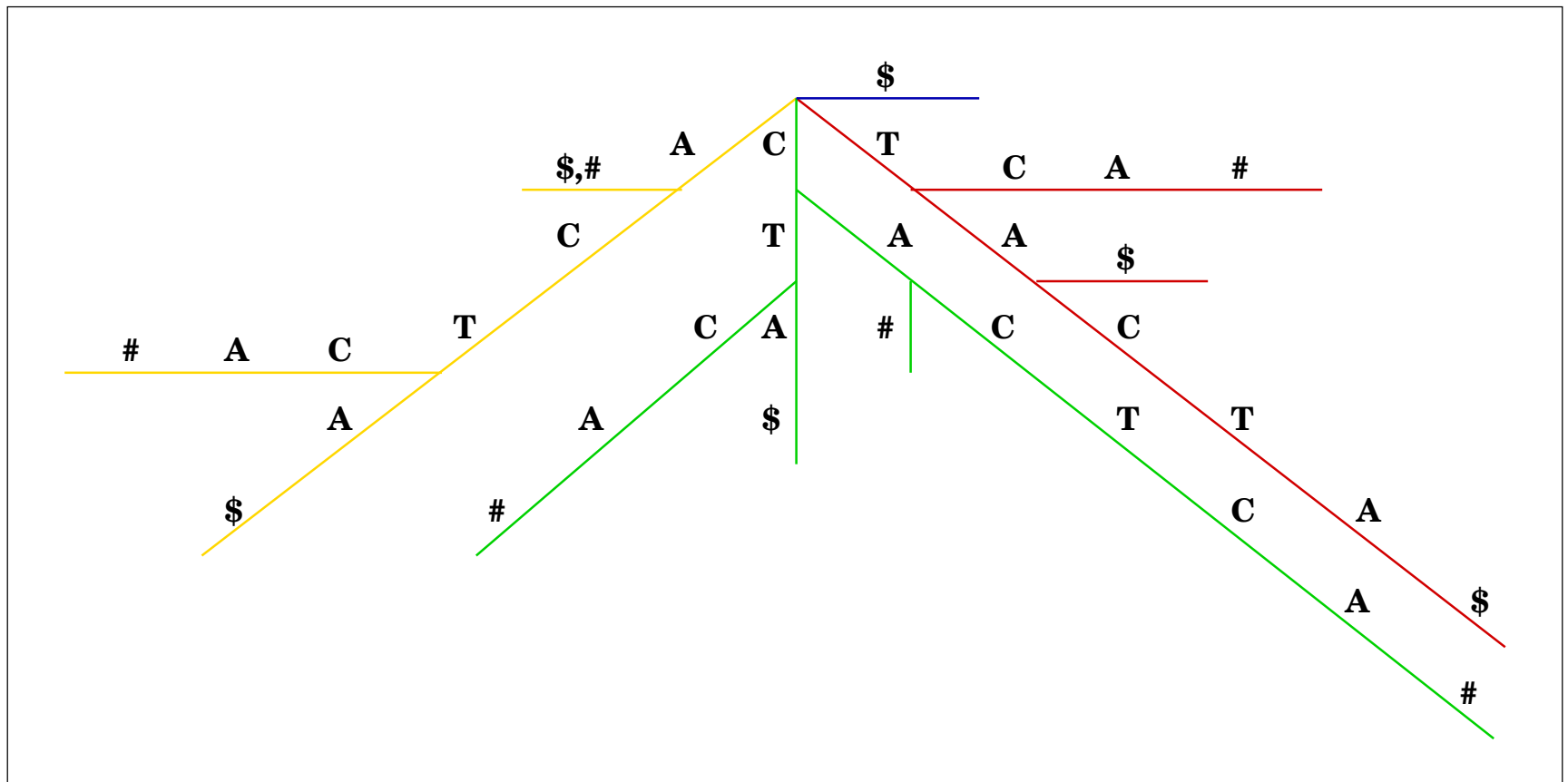
# Generalized Suffix Tree

Suffix tree for the strings TACTA\$ and CACTCA#



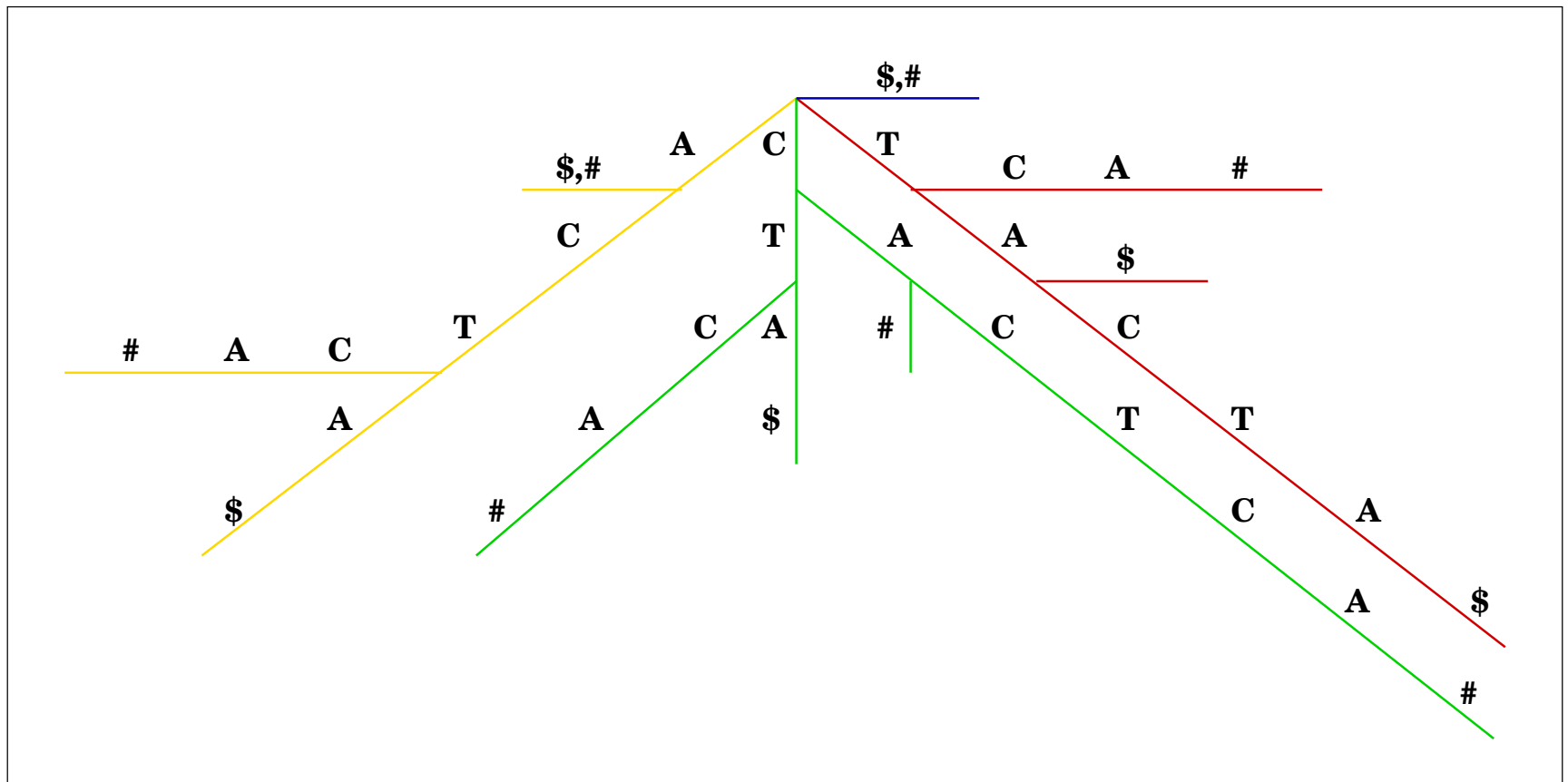
# Generalized Suffix Tree

Suffix tree for the strings TACTA\$ and CACTCA#



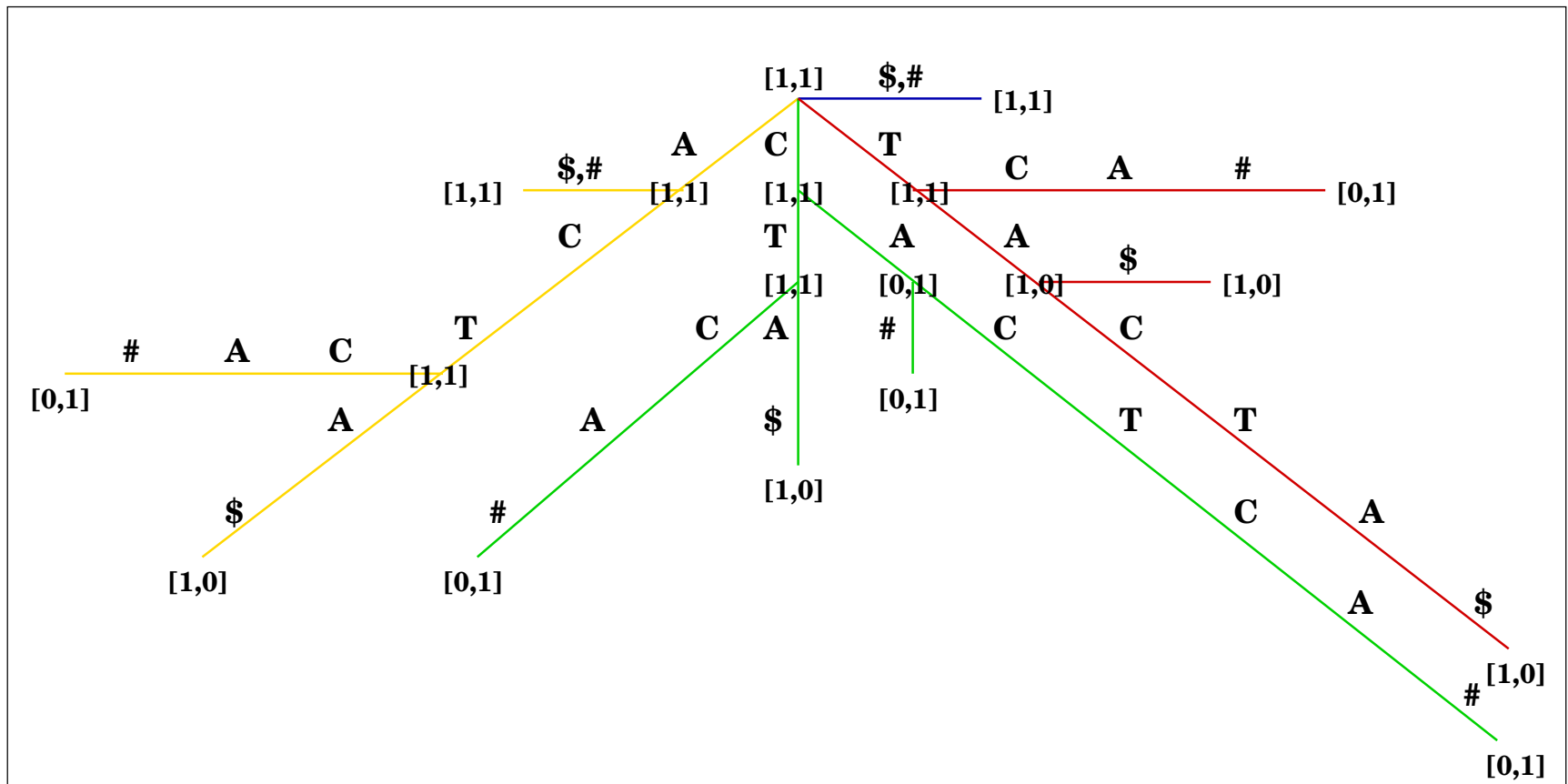
# Generalized Suffix Tree

Suffix tree for the strings TACTA\$ and CACTCA#



# Generalized Suffix Tree with *Colors*

Suffix tree for the strings TACTA\$ and CACTCA#



[ ]: bit vectors called *Colors*



# Extraction of Single Models

**Definition.**  *$e$ -node-occurrence*

A  $e$ -node-occurrence of a model  $m$  is represented by a pair  $(v, e_v)$  where  $v$  is a tree node and  $e_v \leq e$  is the Hamming distance between the label of the path from the root to  $v$  and  $m$ .

**Notation.**  $\nu(e, k)$

The number of distinct words at Hamming distance at most  $e$  from a  $k$ -long word:

$$\nu(e, k) = \sum_{i=0}^e \binom{k}{i} (|\Sigma| - 1)^i \leq k^e |\Sigma|^e.$$

**Notation.**  $n_k$

The number of tree nodes at depth  $k$  of a suffix tree.

# Extraction of Single Models

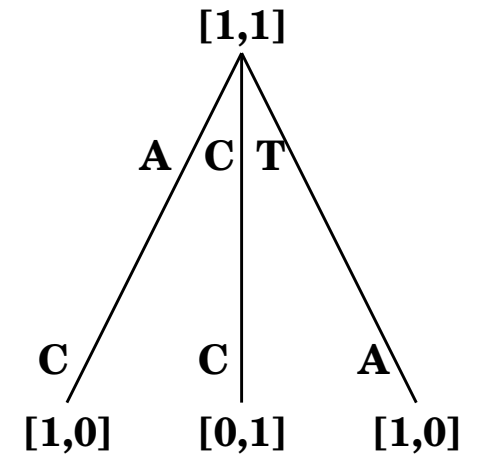
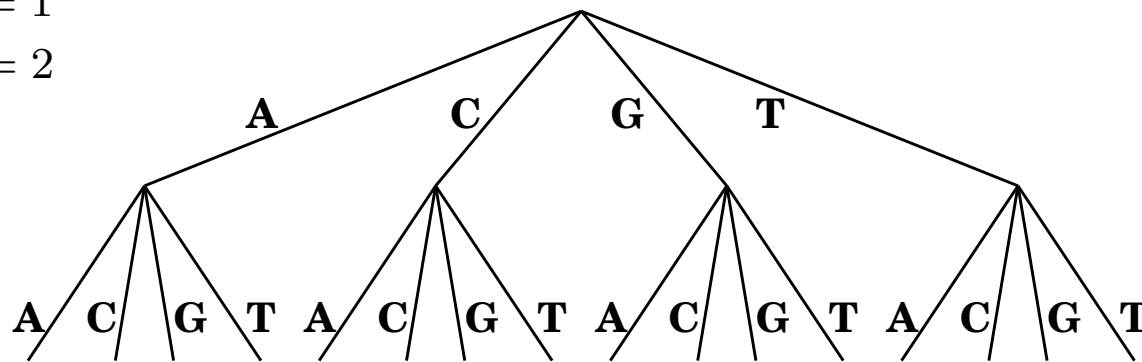
M.-F. Sagot, *Latin*, 1998

$k = 2$

$e = 1$

$q = 2$

Input sequences: TAC and CCC



# Extraction of Single Models

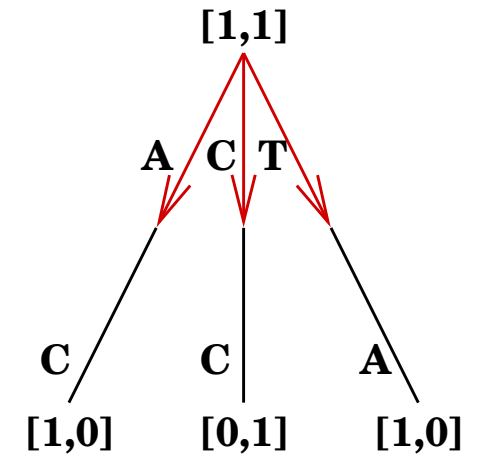
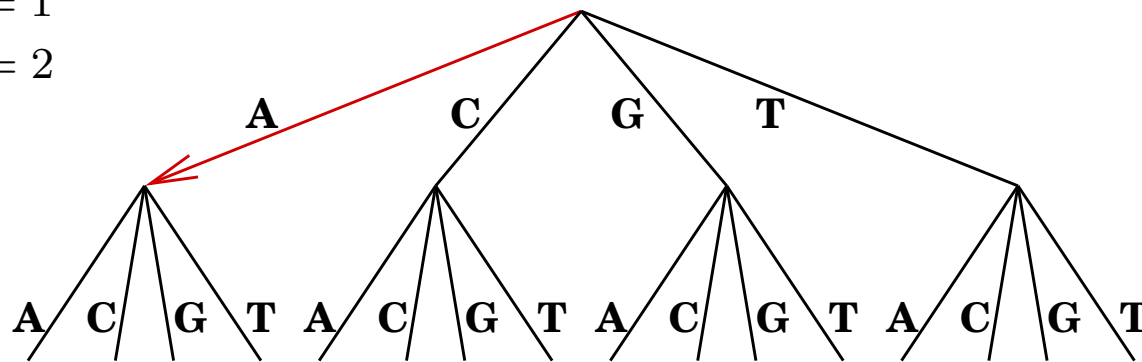
M.-F. Sagot, *Latin*, 1998

$k = 2$

$e = 1$

$q = 2$

Input sequences: TAC and CCC



**(A,0); (C,1); (T,1)**

# Extraction of Single Models

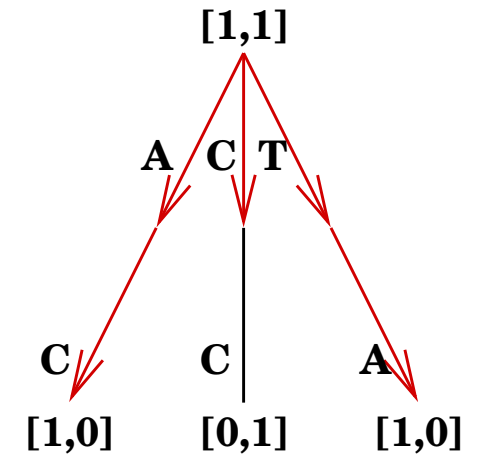
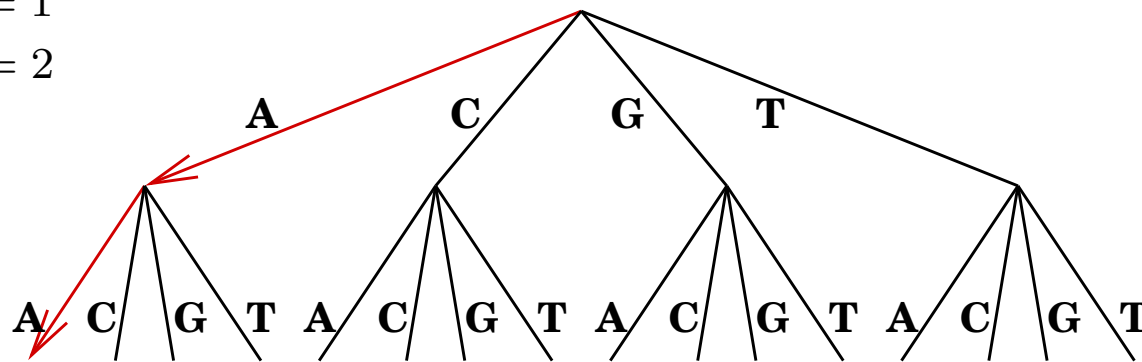
M.-F. Sagot, *Latin*, 1998

$k = 2$

$e = 1$

$q = 2$

Input sequences: TAC and CCC



**(A,0); (C,1); (T,1)**

**(AC,1); (TA,1)**

# Extraction of Single Models

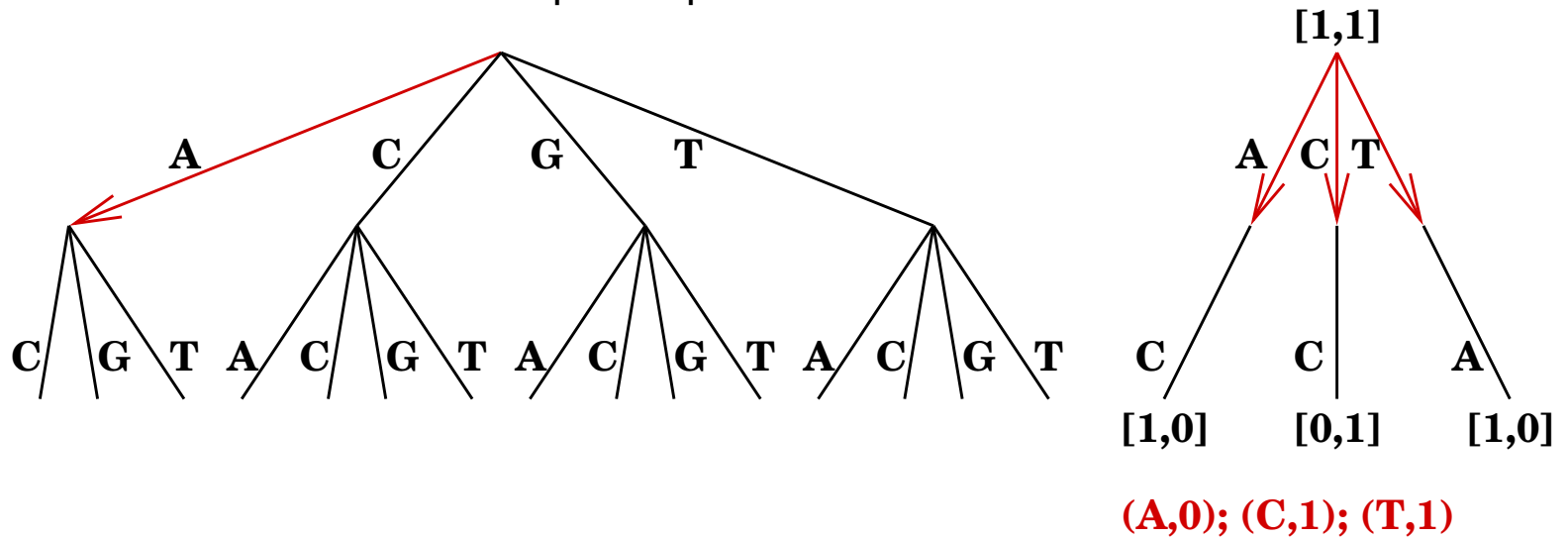
M.-F. Sagot, *Latin*, 1998

$k = 2$

$e = 1$

$q = 2$

Input sequences: TAC and CCC



# Extraction of Single Models

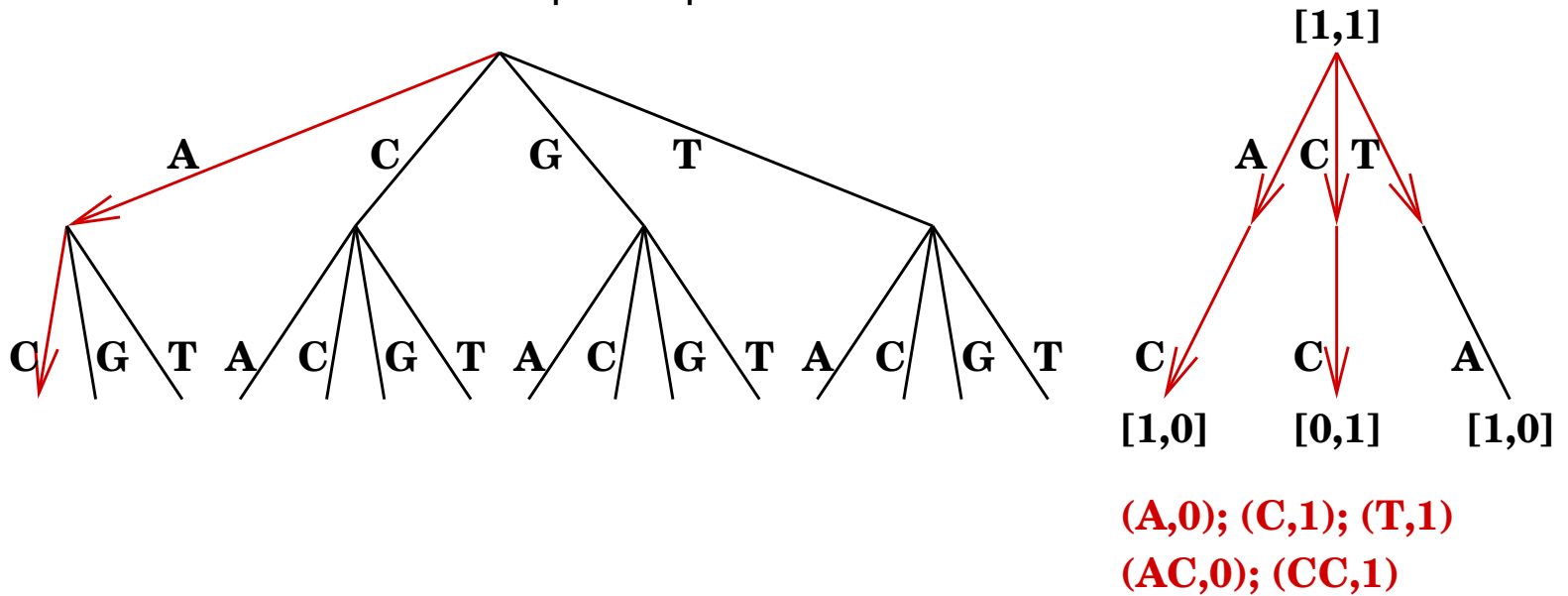
M.-F. Sagot, *Latin*, 1998

$k = 2$

$e = 1$

$q = 2$

Input sequences: TAC and CCC



# Extraction of Single Models

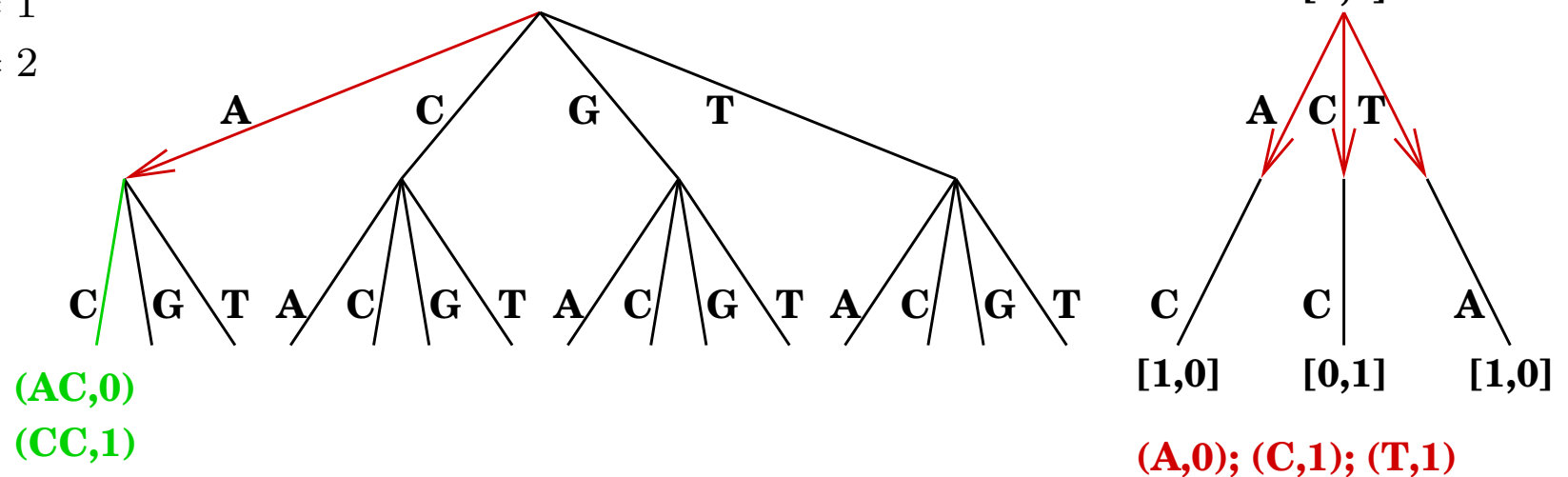
M.-F. Sagot, *Latin*, 1998

$k = 2$

$e = 1$

$q = 2$

Input sequences: TAC and CCC



# Extraction of Single Models

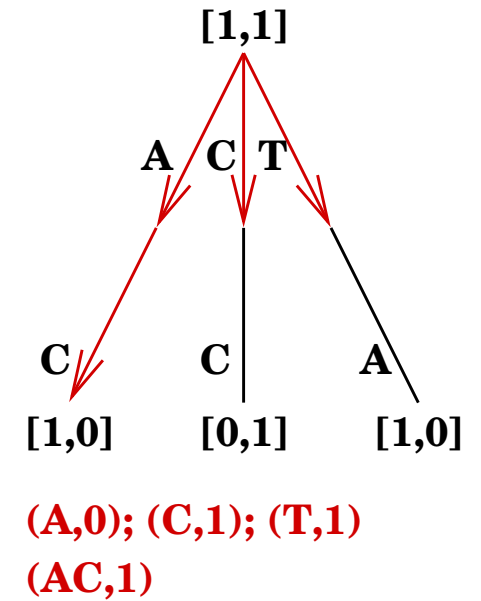
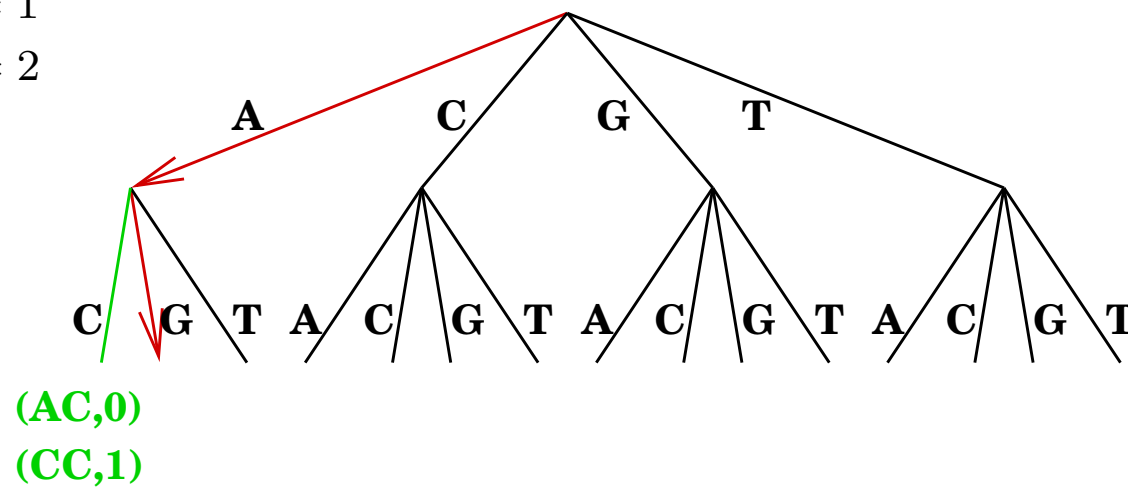
M.-F. Sagot, *Latin*, 1998

$k = 2$

$e = 1$

$q = 2$

Input sequences: TAC and CCC





# Extraction of Single Models

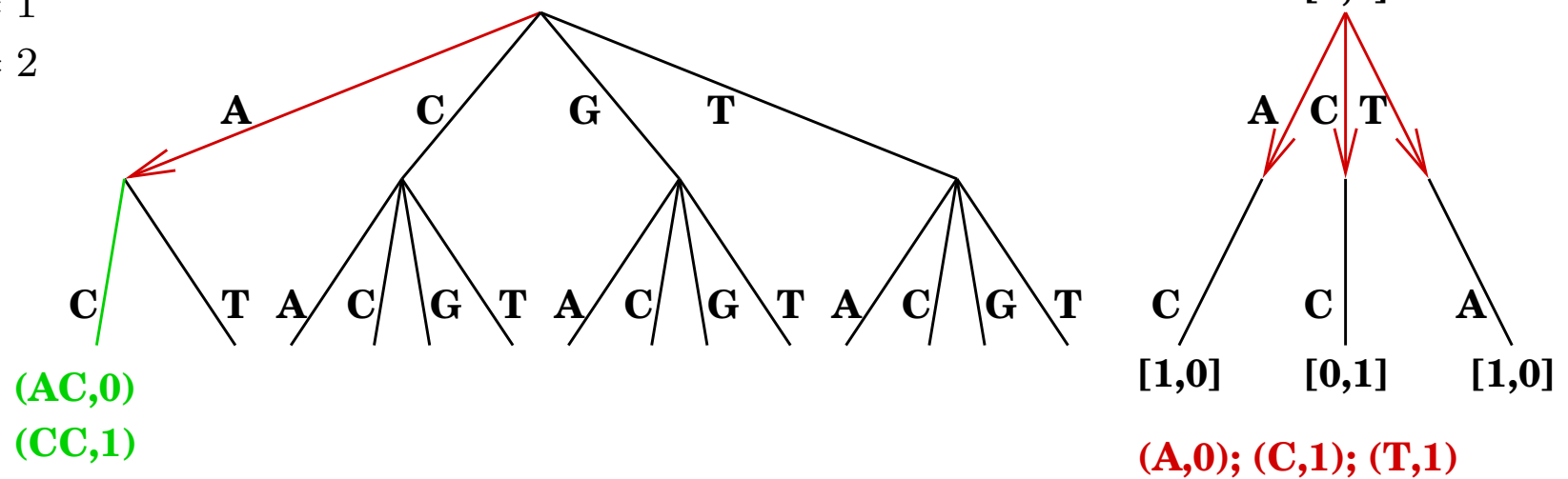
M.-F. Sagot, *Latin*, 1998

$k = 2$

$e = 1$

$q = 2$

Input sequences: TAC and CCC



# Extraction of Single Models

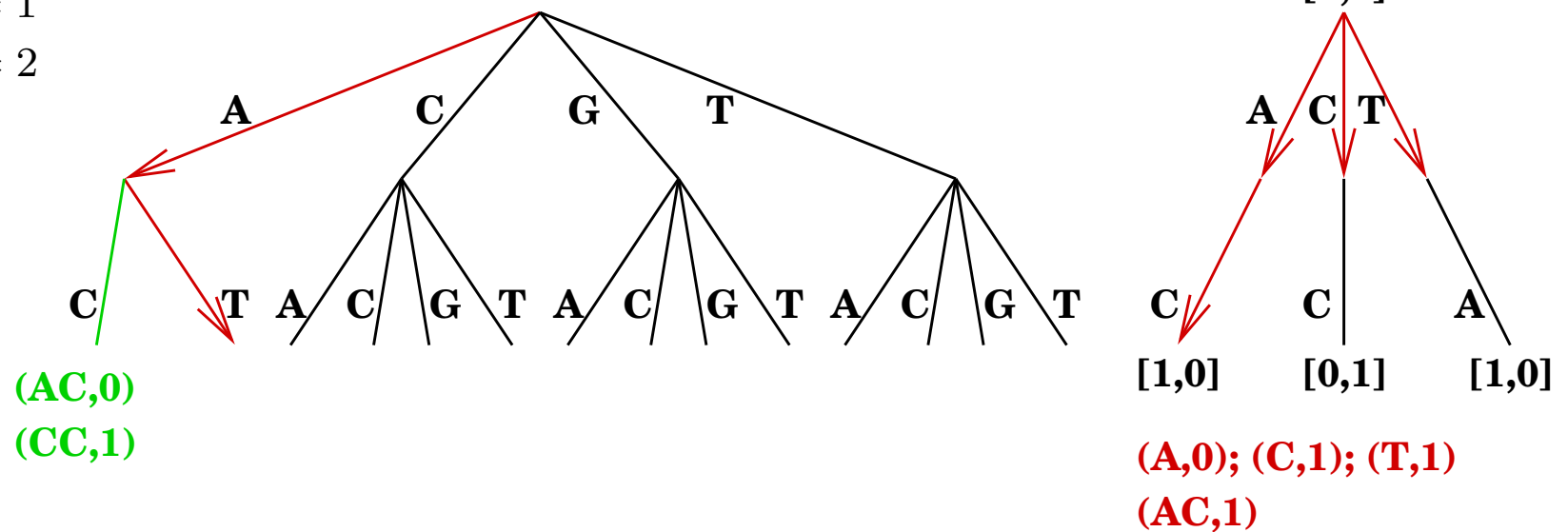
M.-F. Sagot, *Latin*, 1998

$k = 2$

$e = 1$

$q = 2$

Input sequences: TAC and CCC



# Extraction of Single Models

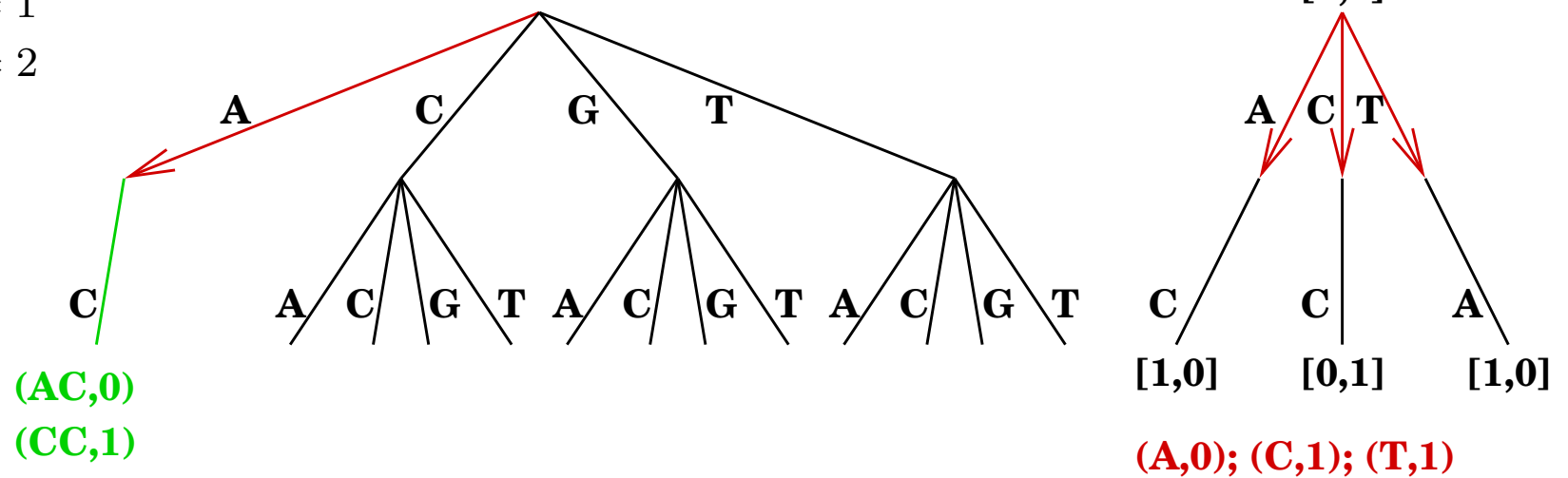
M.-F. Sagot, *Latin*, 1998

$k = 2$

$e = 1$

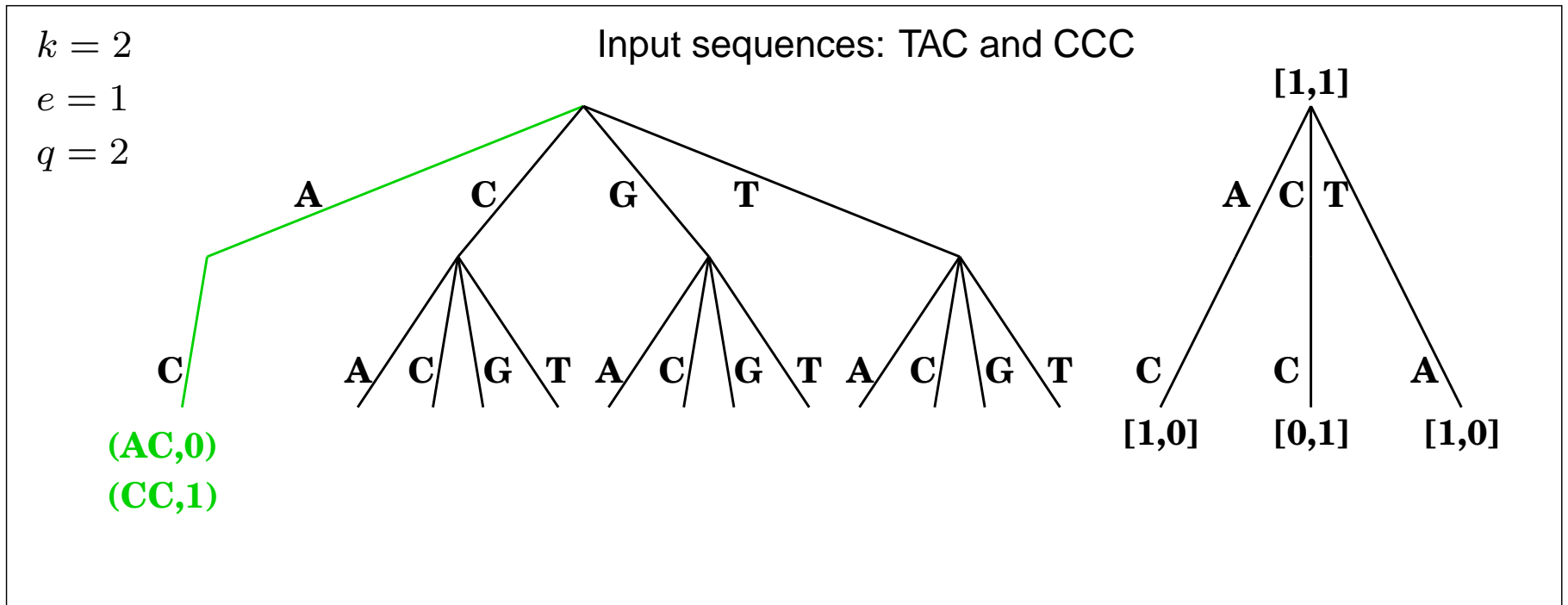
$q = 2$

Input sequences: TAC and CCC



# Extraction of Single Models

M.-F. Sagot, *Latin*, 1998



11



# Extraction of Single Models

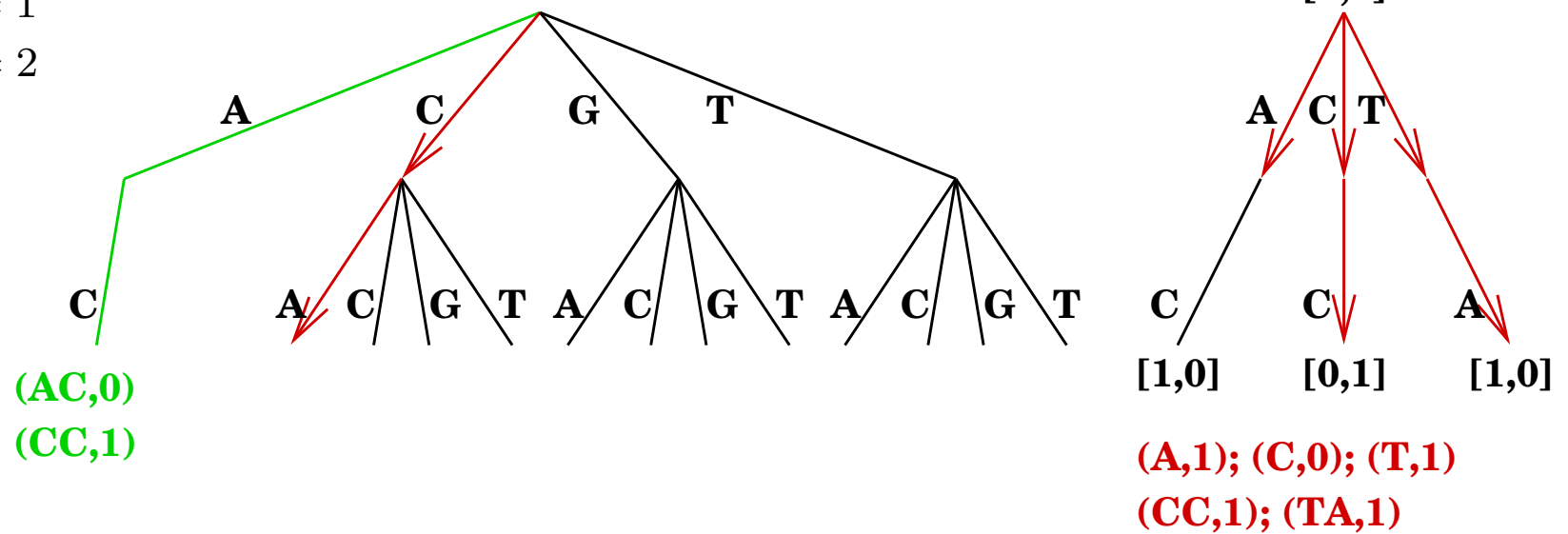
M.-F. Sagot, *Latin*, 1998

$k = 2$

$e = 1$

$q = 2$

Input sequences: TAC and CCC



# Extraction of Single Models

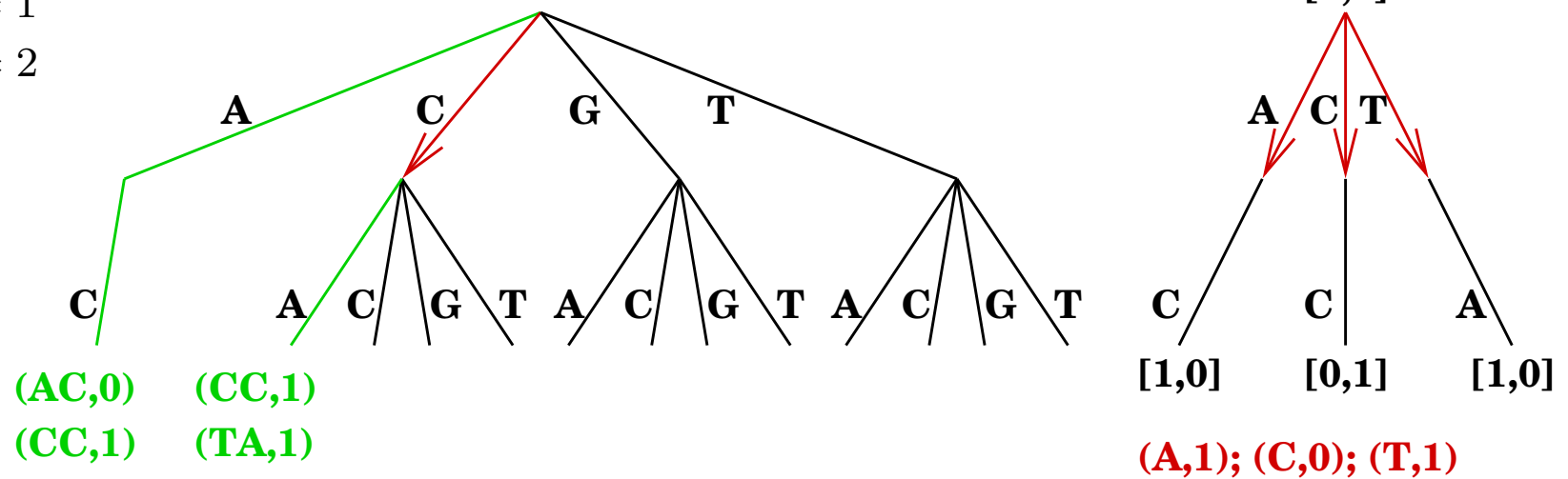
M.-F. Sagot, *Latin*, 1998

$k = 2$

$e = 1$

$q = 2$

Input sequences: TAC and CCC



# Extraction of Single Models

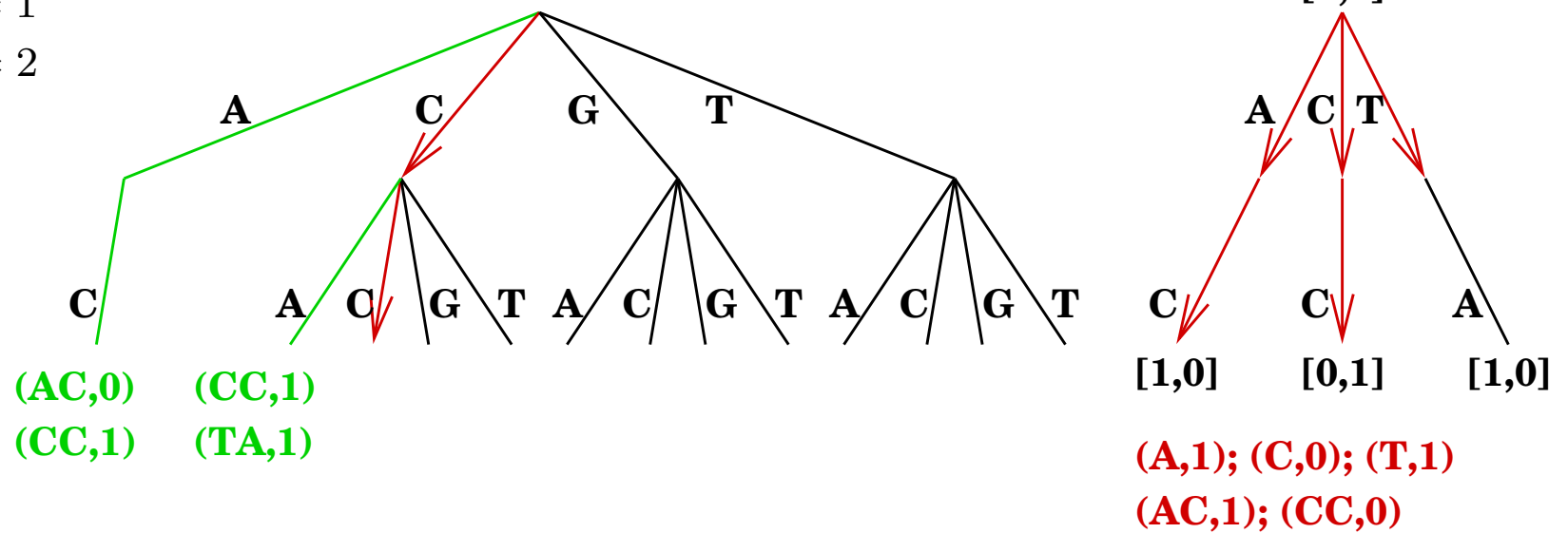
M.-F. Sagot, *Latin*, 1998

$k = 2$

$e = 1$

$q = 2$

Input sequences: TAC and CCC





# Extraction of Single Models

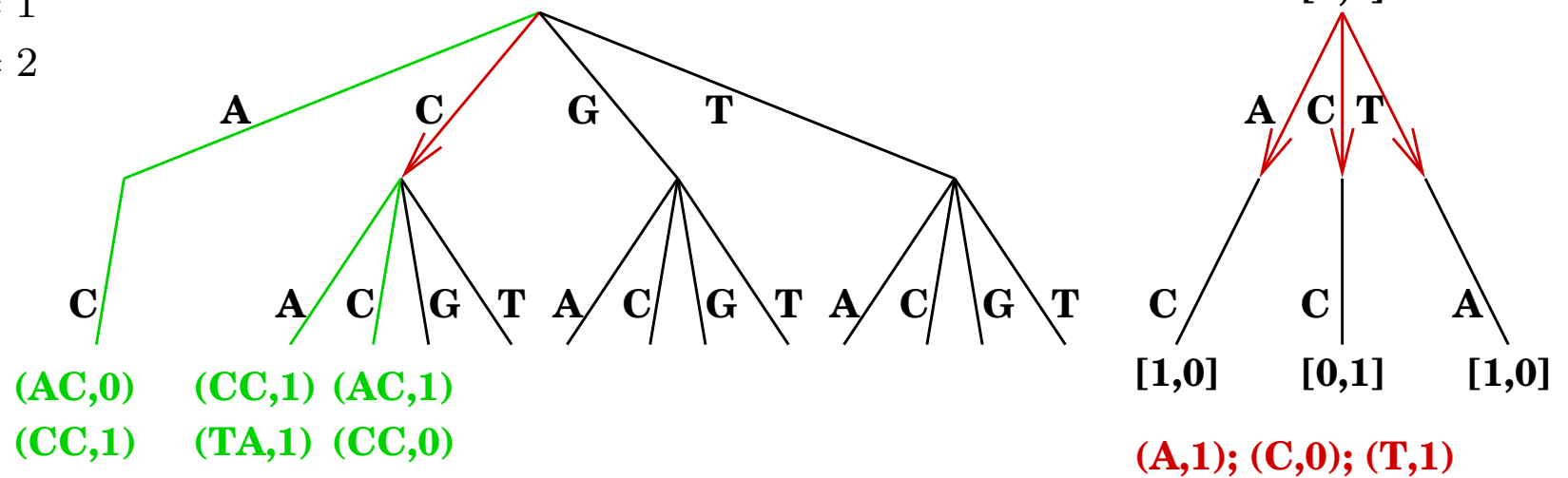
M.-F. Sagot, *Latin*, 1998

$k = 2$

$e = 1$

$q = 2$

Input sequences: TAC and CCC



# Extraction of Single Models

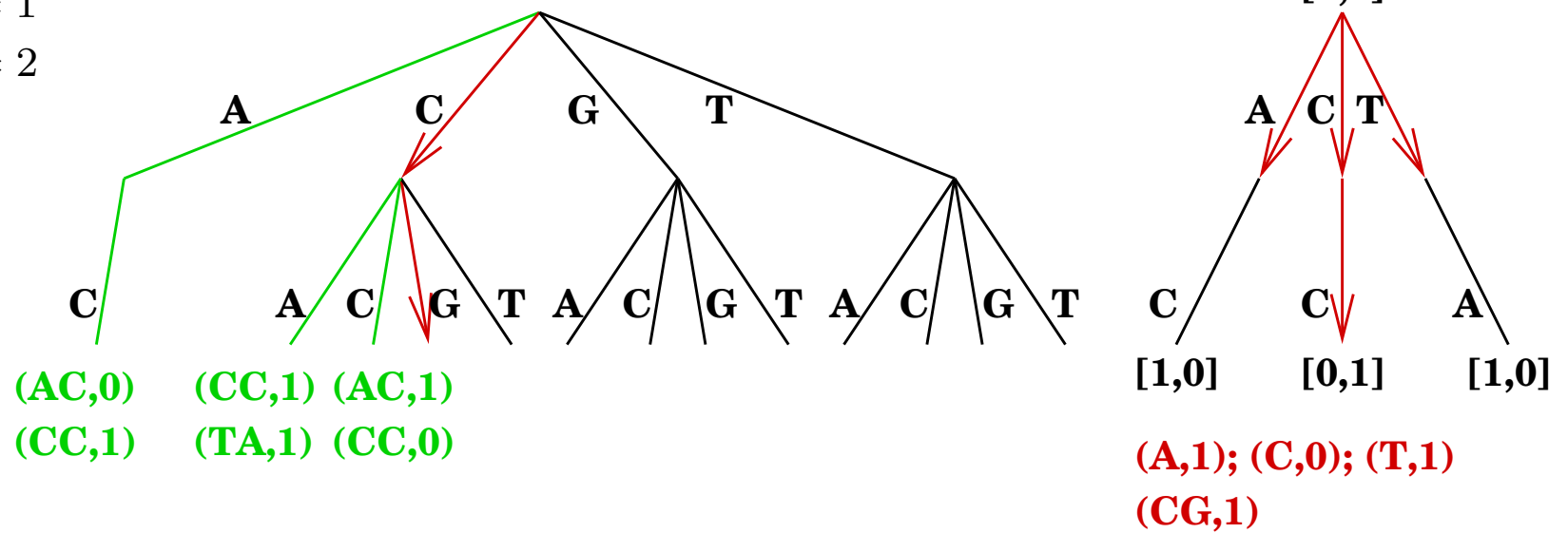
M.-F. Sagot, *Latin*, 1998

$k = 2$

$e = 1$

$q = 2$

Input sequences: TAC and CCC



# Extraction of Single Models

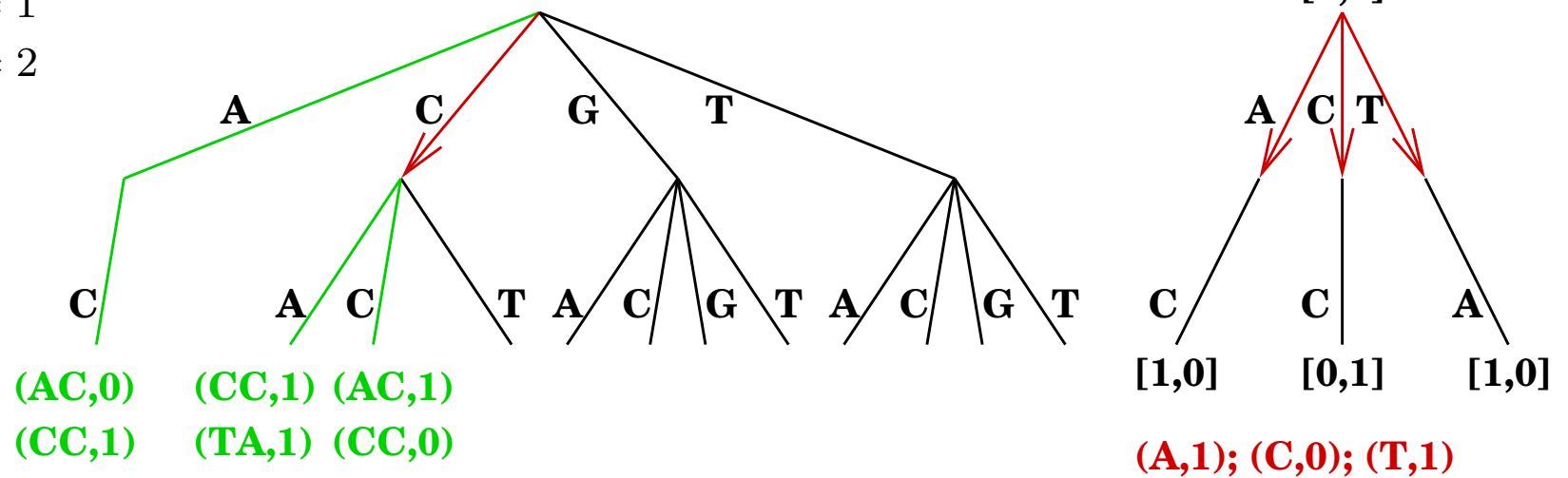
M.-F. Sagot, *Latin*, 1998

$k = 2$

$e = 1$

$q = 2$

Input sequences: TAC and CCC



# Extraction of Single Models

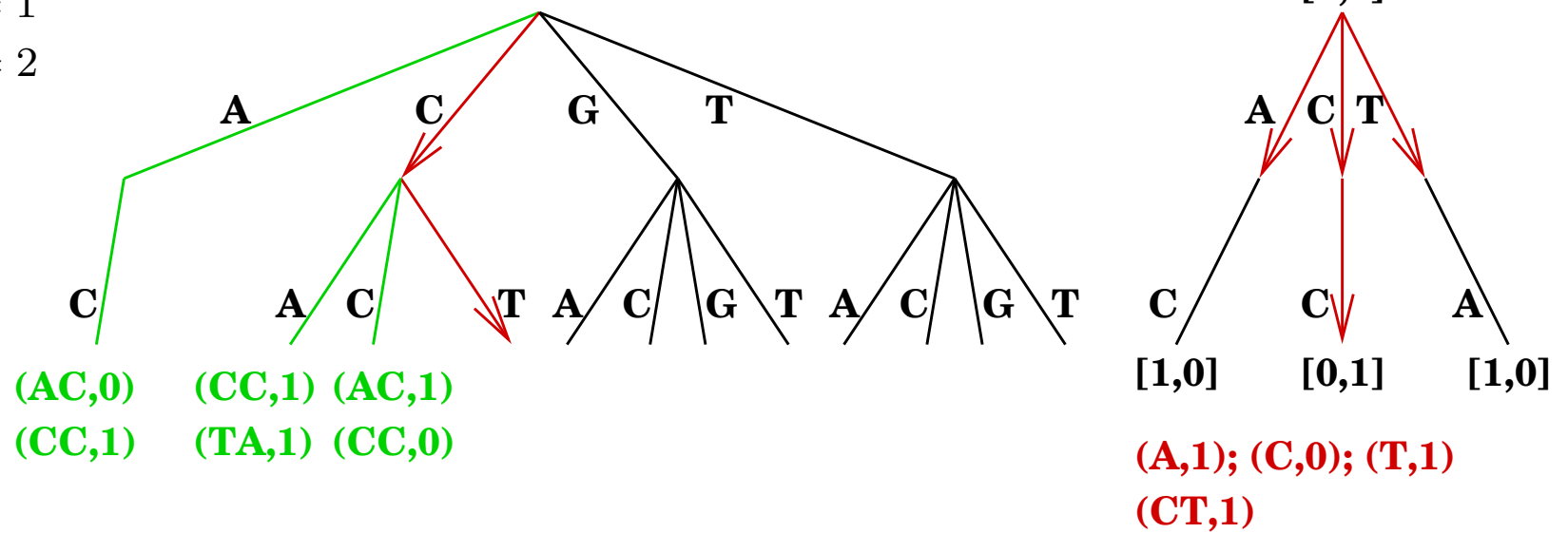
M.-F. Sagot, *Latin*, 1998

$k = 2$

$e = 1$

$q = 2$

Input sequences: TAC and CCC



# Extraction of Single Models

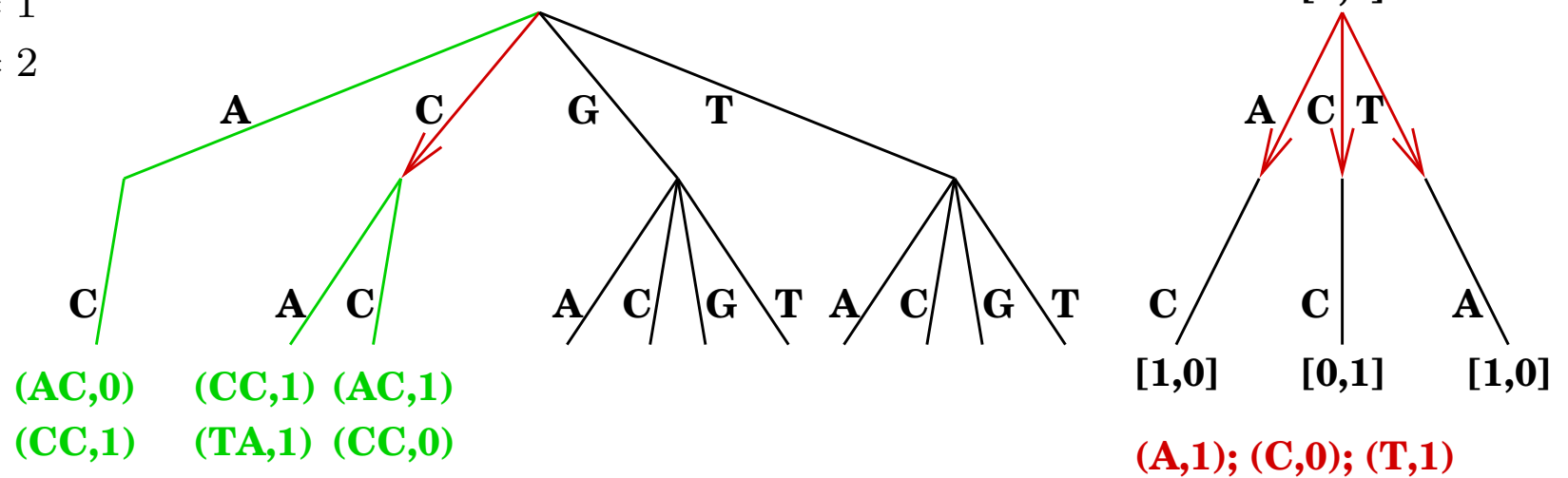
M.-F. Sagot, *Latin*, 1998

$k = 2$

$e = 1$

$q = 2$

Input sequences: TAC and CCC



# Extraction of Single Models

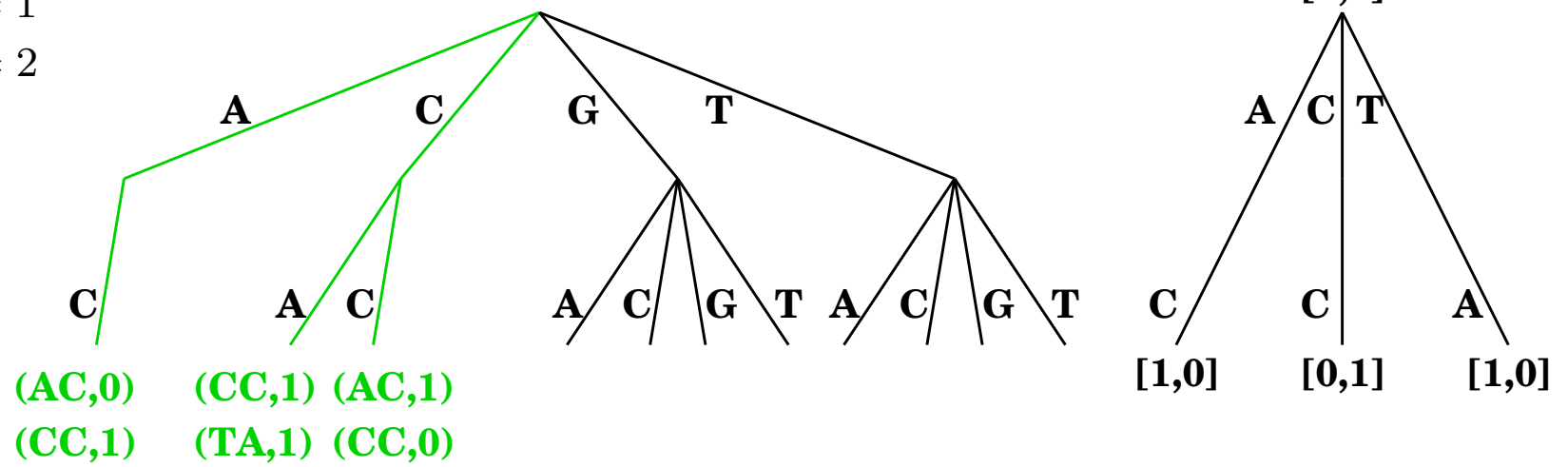
M.-F. Sagot, *Latin*, 1998

$k = 2$

$e = 1$

$q = 2$

Input sequences: TAC and CCC



# Extraction of Single Models

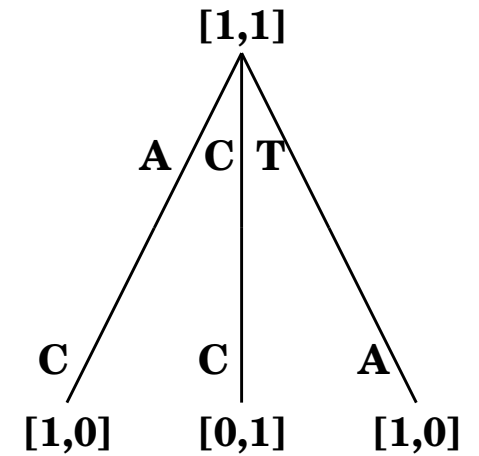
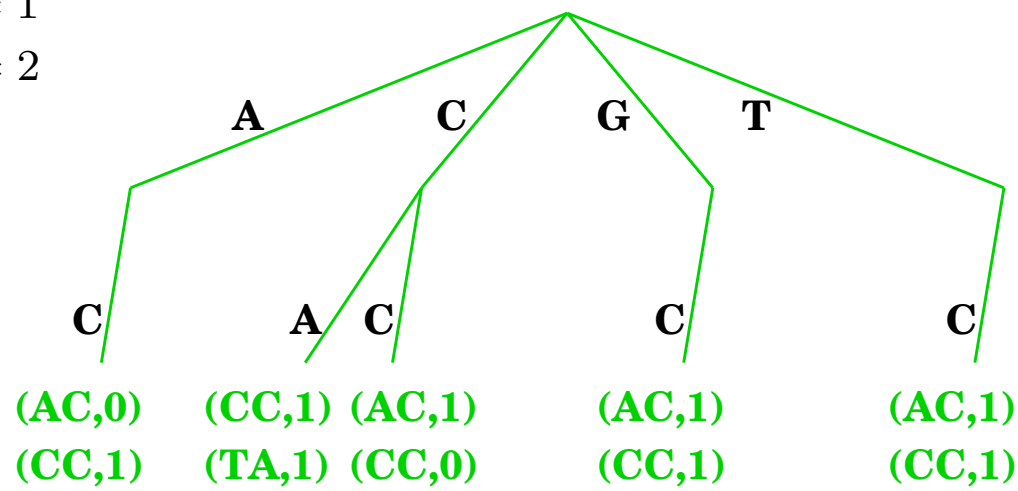
M.-F. Sagot, *Latin*, 1998

$k = 2$

$e = 1$

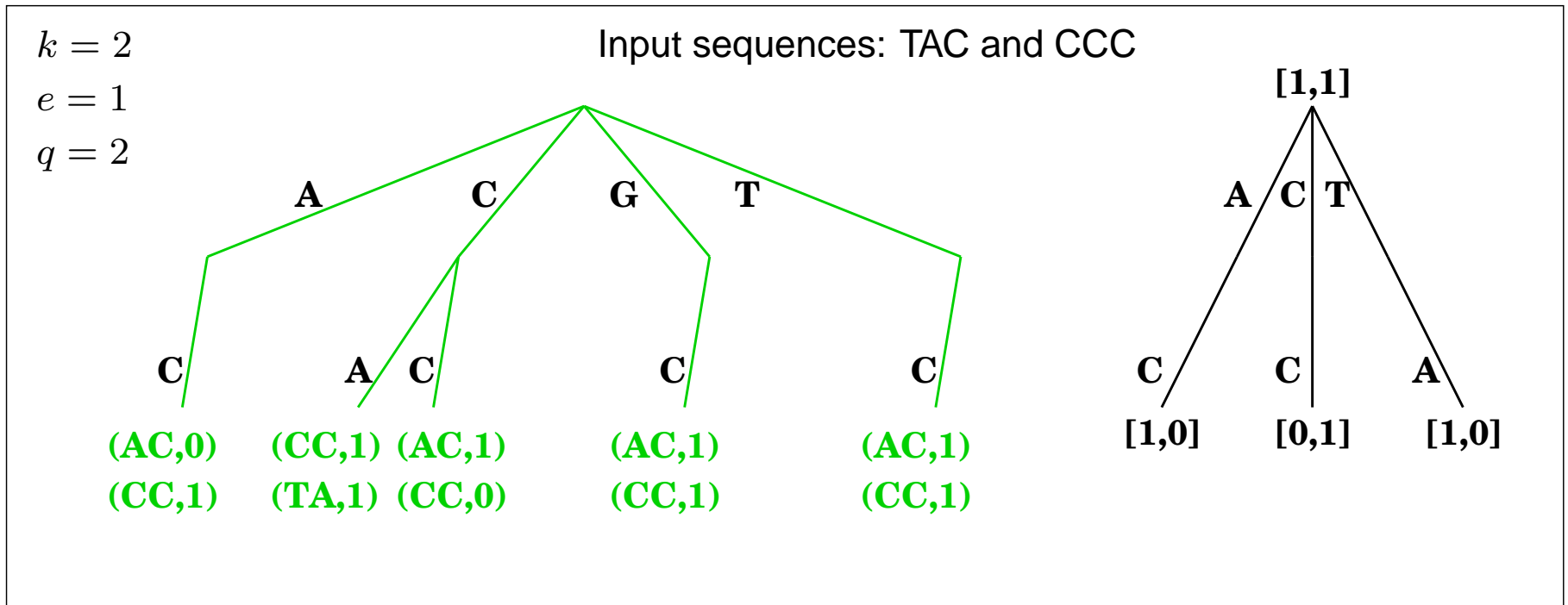
$q = 2$

Input sequences: TAC and CCC



# Extraction of Single Models

M.-F. Sagot, *Latin*, 1998



**Proposition.** The single motifs extraction takes  $O(Nn_k\nu(e, k))$  time.



# Extraction of Structured Models: SMILE

L. Marsan and M.-F. Sagot, *Journal of Computational Biology*, 2000

**ExtractModels** ( **Model**  $m$ , **Block**  $i$  )

1. for each node-occurrence  $v$  of  $m$
2.     if  $(i > 1)$
3.         put in *PotencialStarts* the children of  $v$  at levels  
            $(i - 1)k + (i - 1)d_{min_{i-1}}$  to  $(i - 1)k + (i - 1)d_{max_{i-1}}$
4.     else
5.         put  $v$  in *PotencialStarts*
6. for each model  $m_i$  obtained by doing a recursive depth-first traversal from the root of the virtual model tree  $\mathcal{M}$  while simultaneously traversing  $\mathcal{T}$  from the node-occurrences in *PotencialStarts*
7.     if  $(i < p)$
8.         **ExtractModels** ( $m = m_1 \dots m_i, i + 1$ )
9.     else
10.         **KeepModel** ( $\langle (m_1, \dots, m_p), ((d_{min_1}, d_{max_1}), \dots, (d_{min_p}, d_{max_p})) \rangle$ )

# Extraction of Structured Models: SMILE

L. Marsan and M.-F. Sagot, *Journal of Computational Biology*, 2000

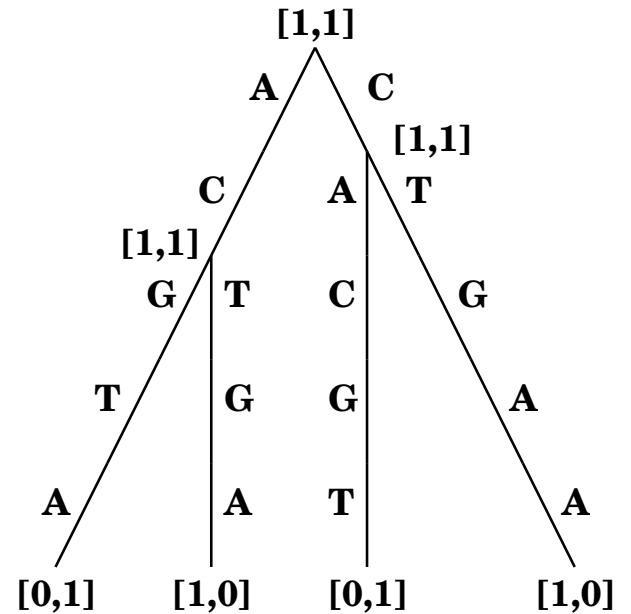
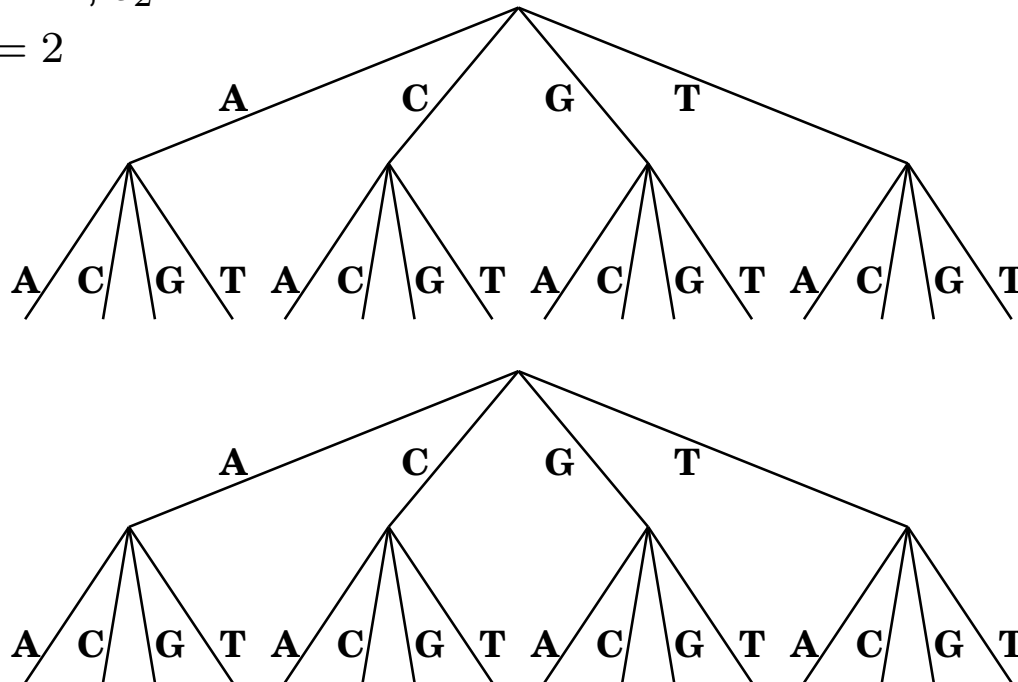
$p = 2$

$k_1 = 2, d = 1, k_2 = 2$

$e_1 = 1, e_2 = 1$

$q = 2$

Input sequences: ACTGAA and CACGTA



# Extraction of Structured Models: SMILE

L. Marsan and M.-F. Sagot, *Journal of Computational Biology*, 2000

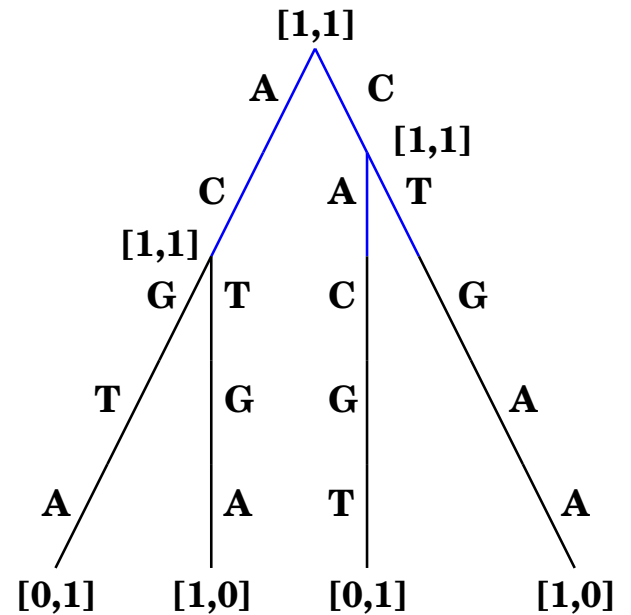
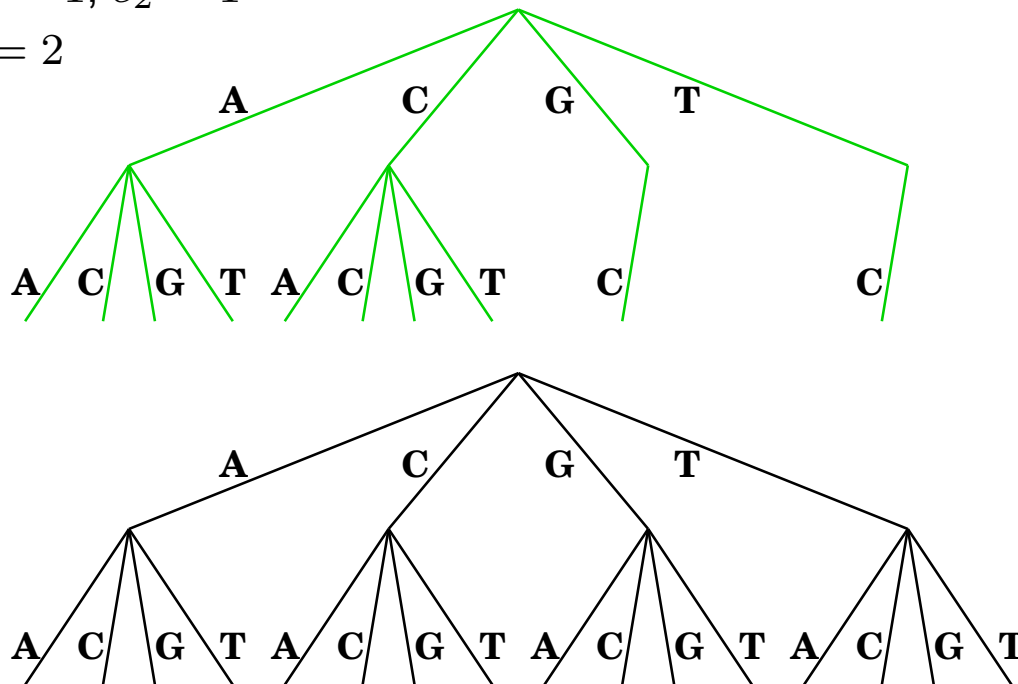
$p = 2$

$k_1 = 2, d = 1, k_2 = 2$

$e_1 = 1, e_2 = 1$

$q = 2$

Input sequences: ACTGAA and CACGTA



# Extraction of Structured Models: SMILE

L. Marsan and M.-F. Sagot, *Journal of Computational Biology*, 2000

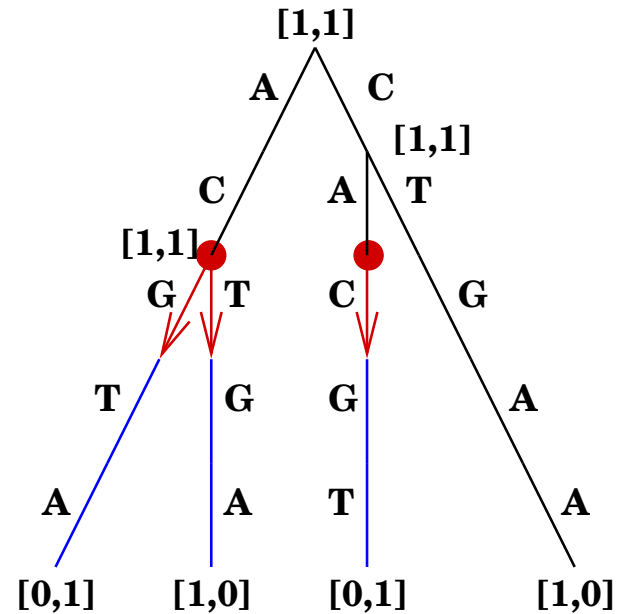
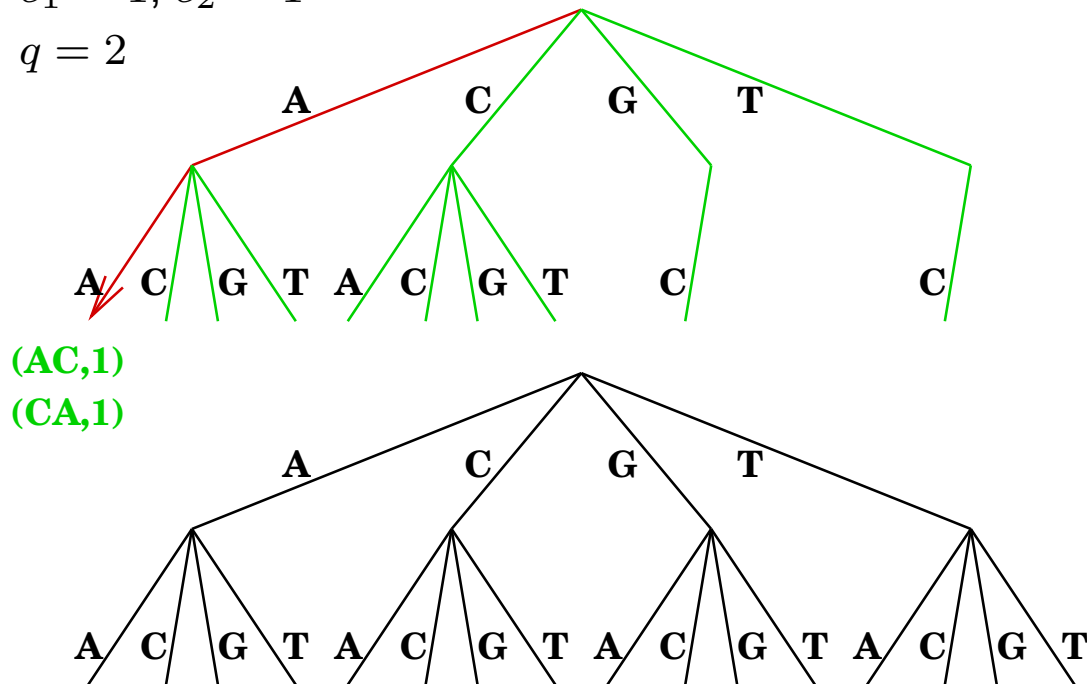
$p = 2$

$k_1 = 2, d = 1, k_2 = 2$

$e_1 = 1, e_2 = 1$

$q = 2$

Input sequences: ACTGAA and CACGTA

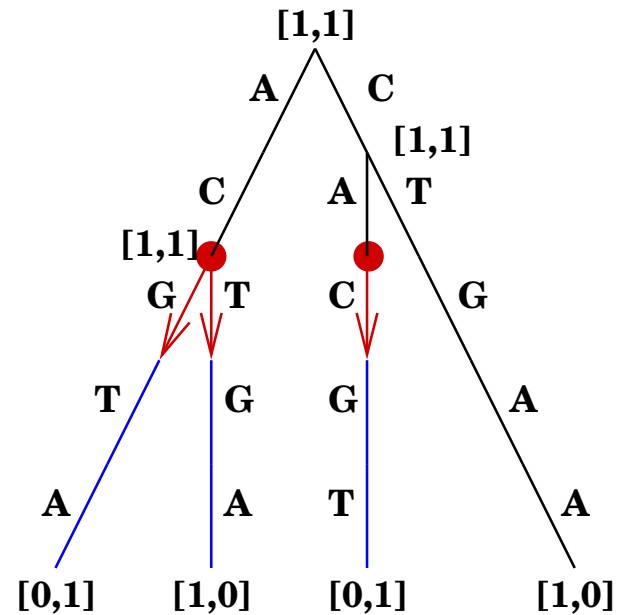
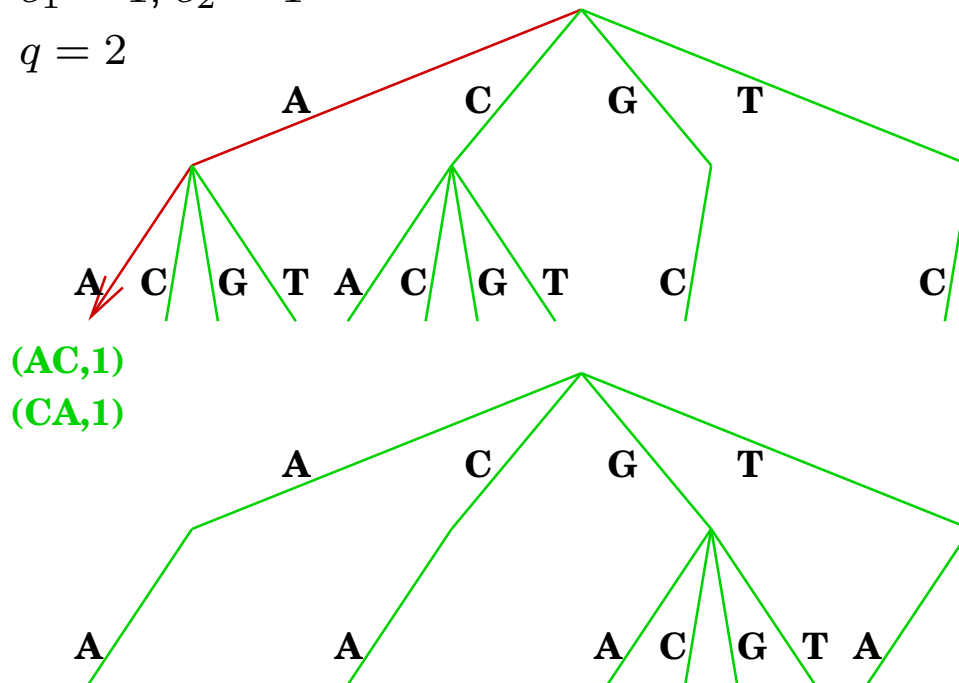


# Extraction of Structured Models: SMILE

**L. Marsan and M.-F. Sagot, *Journal of Computational Biology*, 2000**

$$p = 2$$
$$k_1 = 2, d = 1, k_2 = 2$$
$$e_1 = 1, e_2 = 1$$
$$q = 2$$

Input sequences: ACTGAA and CACGTA



# Extraction of Structured Models: SMILE

L. Marsan and M.-F. Sagot, *Journal of Computational Biology*, 2000

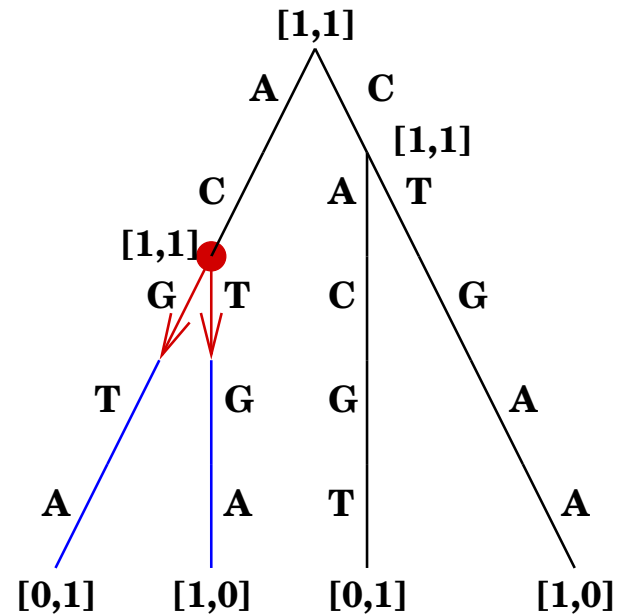
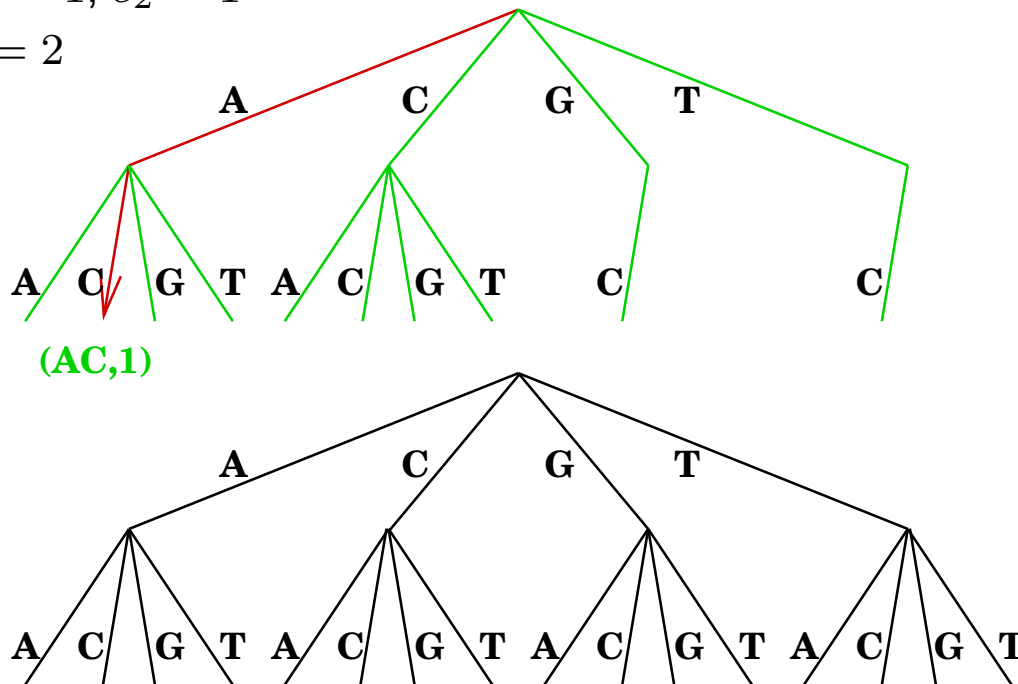
$p = 2$

$k_1 = 2, d = 1, k_2 = 2$

$e_1 = 1, e_2 = 1$

$q = 2$

Input sequences: ACTGAA and CACGTA

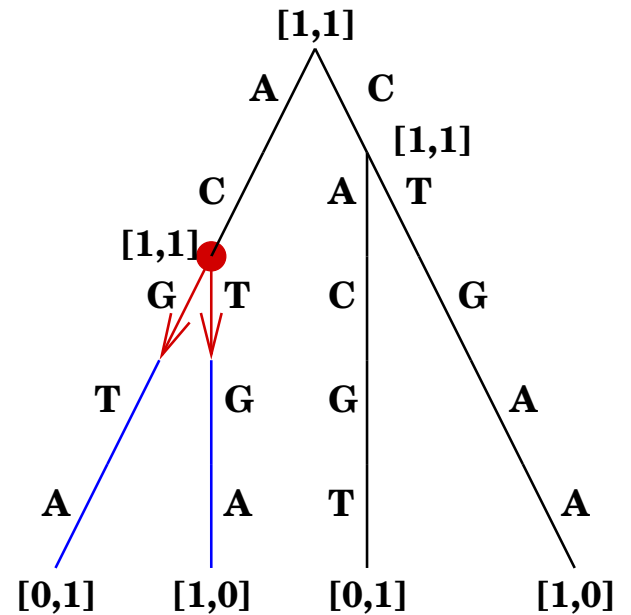
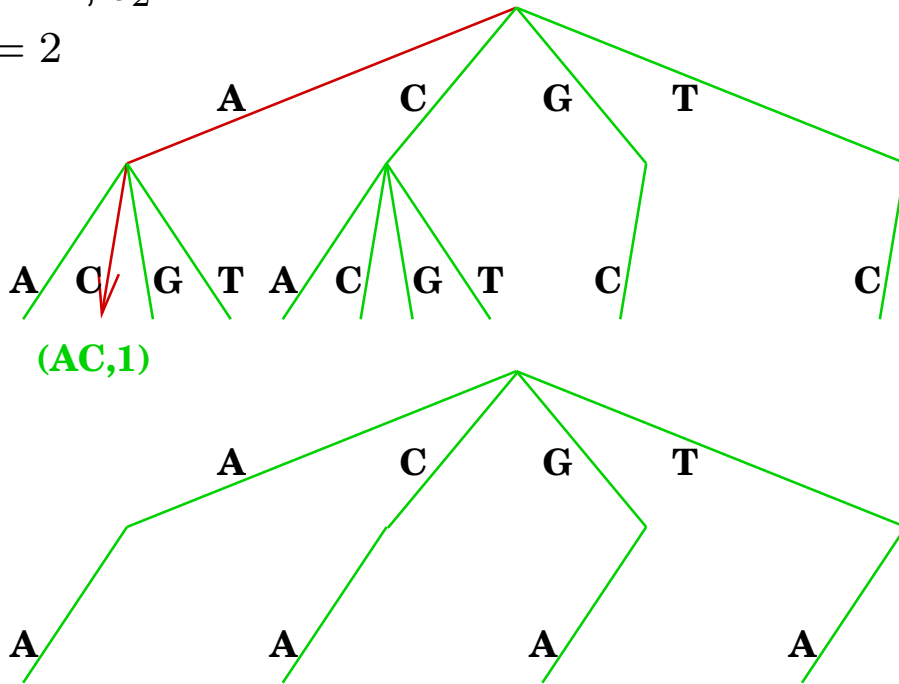


# Extraction of Structured Models: SMILE

**L. Marsan and M.-F. Sagot, *Journal of Computational Biology*, 2000**

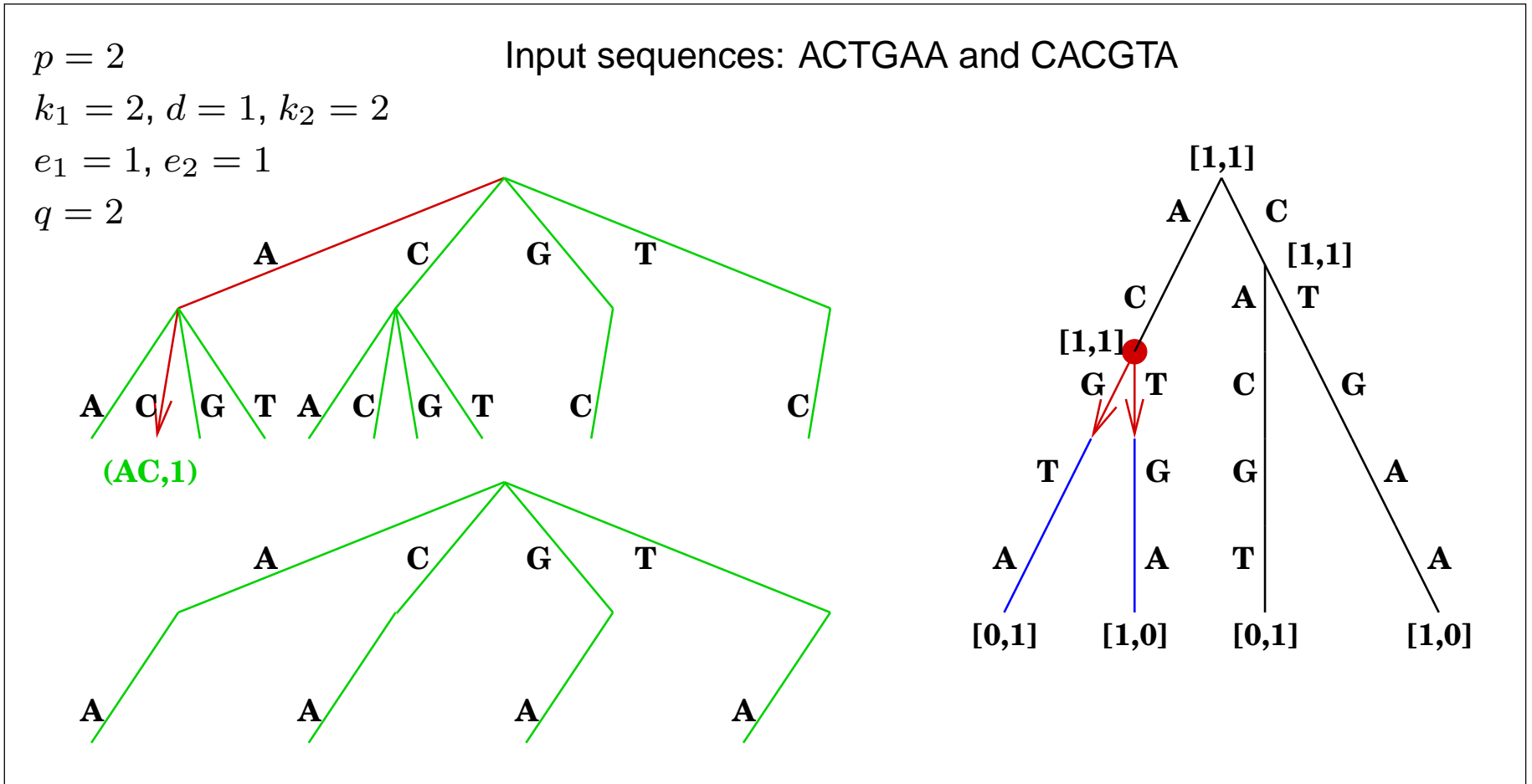
$$p = 2$$
$$k_1 = 2, d = 1, k_2 = 2$$
$$e_1 = 1, e_2 = 1$$
$$q = 2$$

Input sequences: ACTGAA and CACGTA



# Extraction of Structured Models: SMILE

**L. Marsan and M.-F. Sagot, *Journal of Computational Biology*, 2000**



**Proposition.** The structured motifs extraction takes  $O(Nn_{pk+(p-1)d_{max}}\nu^p(e, k))$  time.



# PARTITION UP TO $\varepsilon$

**PARTITION UP TO  $\varepsilon$  problem:**

- $\ell$  gold bars
- $w_i \geq 0$  is the the weight of the  $i$ th gold bar
- any gold bar can be cut in  $c$  equal parts

**Optimization version:** The problem is how to share the gold between  $r$  persons, with the minimum number of gold bars  $z$ , in such a way that each person gets the same share of gold up to some weight  $\varepsilon > 0$ .

**Decision version:** The problem is to decide whether it is possible to share the gold between  $r$  persons, with  $z$  gold bars, in such a way that each person gets the same share of gold up to some weight  $\varepsilon \geq 0$ .

**Proposition.** The PARTITION UP TO  $\varepsilon$  problem is NP-complete in the strong sense.

# PARTITION UP TO $\varepsilon$

**SimpleCut** ( Partition  $i$ , GoldBars  $\ell$ , Persons  $r$ , Weights  $w_j$ , CutFactor  $c$ , WorkOverload  $\varepsilon$  )

1. find the smallest  $t$  such that  $\frac{\max w_j}{c^t} \leq \varepsilon$
2. for each  $j \in \{1, \dots, \ell\}$
3.     let  $V_j = \left[ \sum_{k=1}^{j-1} w_k \times c^t, \sum_{k=1}^j w_k \times c^t \right)$
4. let  $w = \sum_{j=1}^{\ell} w_j$
5. let  $\gamma = w \times c^t \bmod r$
6. let  $\delta = \lfloor \frac{w \times c^t}{r} \rfloor$
7. let  $I'_i = \begin{cases} [(i-1)(\delta+1), i(\delta+1)) & \text{for all } i \leq \gamma \\ [\gamma(\delta+1) + (i - (\gamma+1))\delta, \gamma(\delta+1) + (i - \gamma)\delta) & \text{otherwise} \end{cases}$
8. transform  $I'_i = [a, b)$  into  $I_i = [f(a), f(b))$  with  $f: w \times c^t \rightarrow \ell \times c^t$ :

$$f(x) = \begin{cases} (j-1) \times c^t + \frac{x - \inf(V_j)}{w_j} & \text{for all } x \in V_j \\ \ell \times c^t & \text{if } x = w \times c^t \end{cases}$$

# PARTITION UP TO $\varepsilon$

$j$	1	2
$w_j$	2	1

$$r = 3 \quad \varepsilon = 1$$

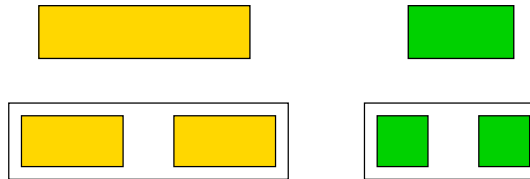
# PARTITION UP TO $\varepsilon$

$j$	1	2
$w_j$	2	1

$r = 3$   $\varepsilon = 1$

$t = 1$

1. find the smallest  $t$  such that  $\frac{\max w_j}{c^t} \leq \varepsilon$



# PARTITION UP TO $\varepsilon$

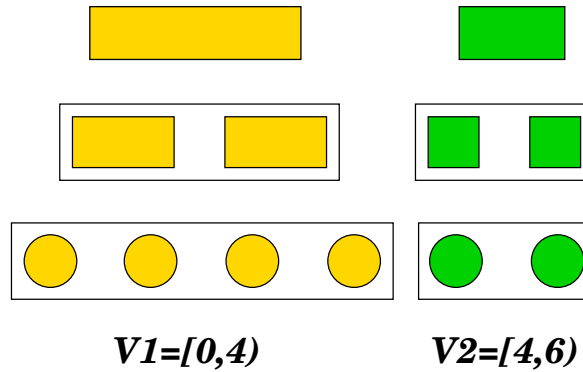
$j$	1	2
$w_j$	2	1

$r = 3 \quad \varepsilon = 1$

$t = 1$

2. for each  $j \in 1, \dots, \ell$

3.  $V_j = \left[ \sum_{k=1}^{j-1} w_k \times c^t, \sum_{k=1}^j w_k \times c^t \right)$



# PARTITION UP TO $\varepsilon$

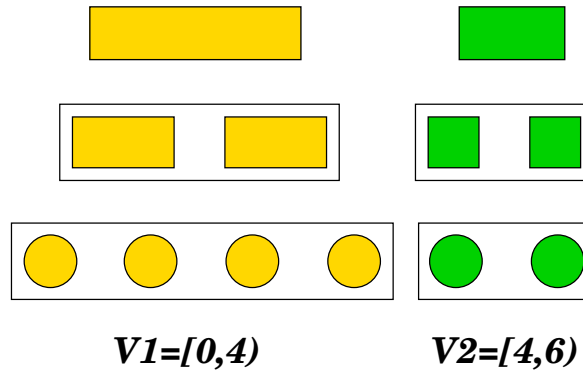
$j$	1	2
$w_j$	2	1

$r = 3 \quad \varepsilon = 1$

$t = 1$

$w = 3 \quad \gamma = 0 \quad \delta = 2$

4.  $w = \sum_{j=1}^{\ell} w_j$
5.  $\gamma = w \times c^t \bmod r$
6.  $\delta = \lfloor \frac{w \times c^t}{r} \rfloor$



# PARTITION UP TO $\varepsilon$

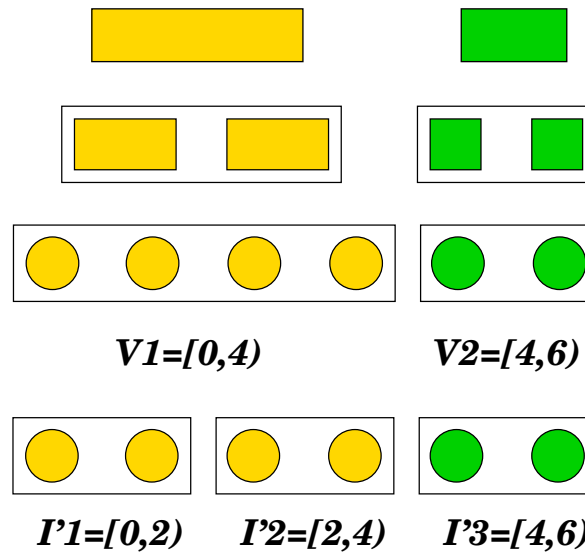
$j$	1	2
$w_j$	2	1

$$7. I'_i = \begin{cases} [(i-1)(\delta+1), i(\delta+1)) & \text{for all } i \leq \gamma \\ [\gamma(\delta+1) + (i-(\gamma+1))\delta, \gamma(\delta+1) + (i-\gamma)\delta) & \text{otherwise} \end{cases}$$

$r = 3 \quad \varepsilon = 1$

$t = 1$

$w = 3 \quad \gamma = 0 \quad \delta = 2$



# PARTITION UP TO $\varepsilon$

$j$	1	2
$w_j$	2	1

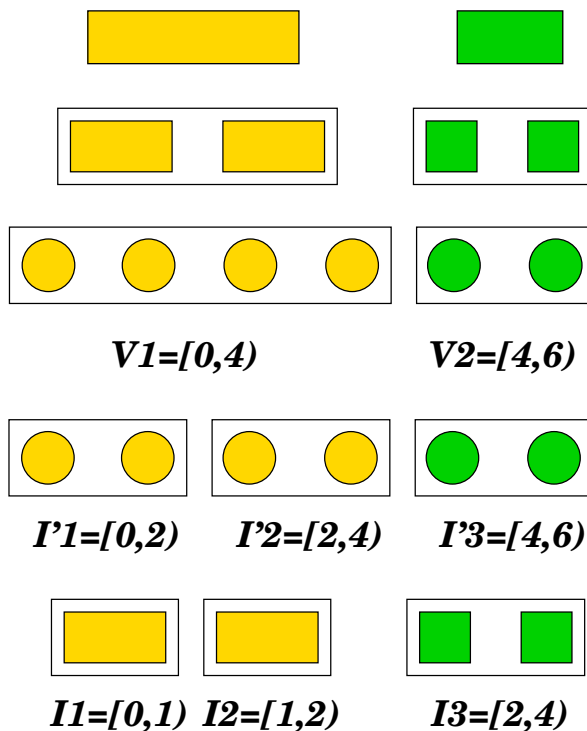
$$r = 3 \quad \varepsilon = 1$$

$$t = 1$$

$$w = 3 \quad \gamma = 0 \quad \delta = 2$$

8. transform  $I'_i = [a, b)$  into  $I_i = [f(a), f(b))$  with

$$f(x) = \begin{cases} (j-1) \times c^t + \frac{x - \inf(V_j)}{w_j} & \text{for all } x \in V_j \\ \ell \times c^t & \text{if } x = w \times c^t \end{cases}$$





# PARTITION UP TO $\varepsilon$

**Proposition.** The SimpleCut algorithm requires  $O(\ell)$  time.

**Proposition.** The SimpleCut algorithm has a ratio bound  $\rho(\ell, r, (w_i)_{1 \leq i \leq \ell}, c, \varepsilon) = O(\frac{\max w_i}{\varepsilon})$ .

# Parallelization

Reducing the tree partition problem to the PARTITION UP TO  $\varepsilon$  problem

Input of the SimpleCut algorithm for the  $i$ th grid node:

- $\ell = |\Sigma|$
- $r$  matches the number of grid nodes
- $w_j$  of each symbol of the alphabet is obtained by scanning the input sequences
- $c = |\Sigma|$
- $\varepsilon$  is an user parameter

# Parallelization

Reducing the tree partition problem to the PARTITION UP TO  $\varepsilon$  problem

Input of the SimpleCut algorithm for the  $i$ th grid node:

- $\ell = |\Sigma|$
- $r$  matches the number of grid nodes
- $w_j$  of each symbol of the alphabet is obtained by scanning the input sequences
- $c = |\Sigma|$
- $\varepsilon$  is a user parameter

Output of the SimpleCut algorithm for the  $i$ th grid node:

- the number  $t$  of cuts gives the depth  $t + 1$  of the tree where the partition is defined
- an interval  $I_i$  corresponding to tree nodes at depth  $t + 1$  assigned to the  $i$ th grid node

# Parallelization

Reducing the tree partition problem to the PARTITION UP TO  $\varepsilon$  problem

Input of the SimpleCut algorithm for the  $i$ th grid node:

- $\ell = |\Sigma|$
- $r$  matches the number of grid nodes
- $w_j$  of each symbol of the alphabet is obtained by scanning the input sequences
- $c = |\Sigma|$
- $\varepsilon$  is a user parameter

Output of the SimpleCut algorithm for the  $i$ th grid node:

- the number  $t$  of cuts gives the depth  $t + 1$  of the tree where the partition is defined
- an interval  $I_i$  corresponding to tree nodes at depth  $t + 1$  assigned to the  $i$ th grid node

**SimpleCut** ( **Partition**  $i$ , **AlphabetSize**  $\ell$ , **GridNodes**  $r$ , **Weights**  $w_j$ , **AlphabetSize**  $c$ , **WorkOverload**  $\varepsilon$  )

1. find the smallest  $t'$  such that  $\frac{\max w_j}{c^{t'}} \leq \varepsilon$
2. let  $t = \min(\text{depth}(\mathcal{M}) - 1, t')$

# Parallelization

$j$	1	2	3	4
$\sigma_j$	A	C	T	G
$w_j$	2	1	1	2

$$r = 5 \quad \varepsilon = 1$$

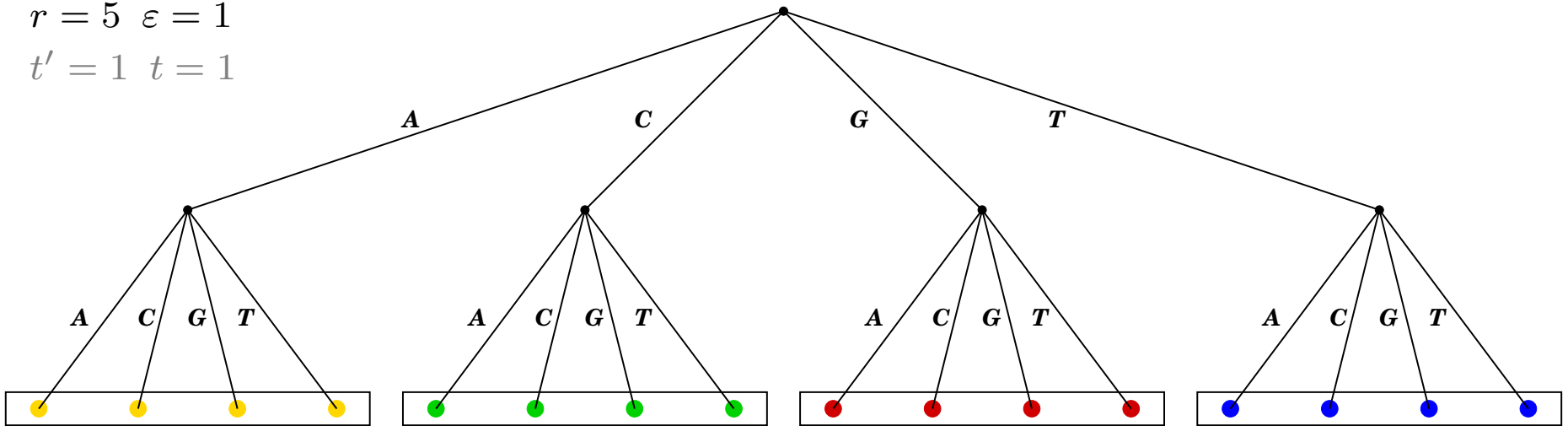
# Parallelization

$j$	1	2	3	4
$\sigma_j$	A	C	T	G
$w_j$	2	1	1	2

1. find the smallest  $t'$  such that  $\frac{\max w_j}{c^{t'}} \leq \varepsilon$
2.  $t = \min(\text{depth}(\mathcal{M}) - 1, t')$

$r = 5 \quad \varepsilon = 1$

$t' = 1 \quad t = 1$



# Parallelization

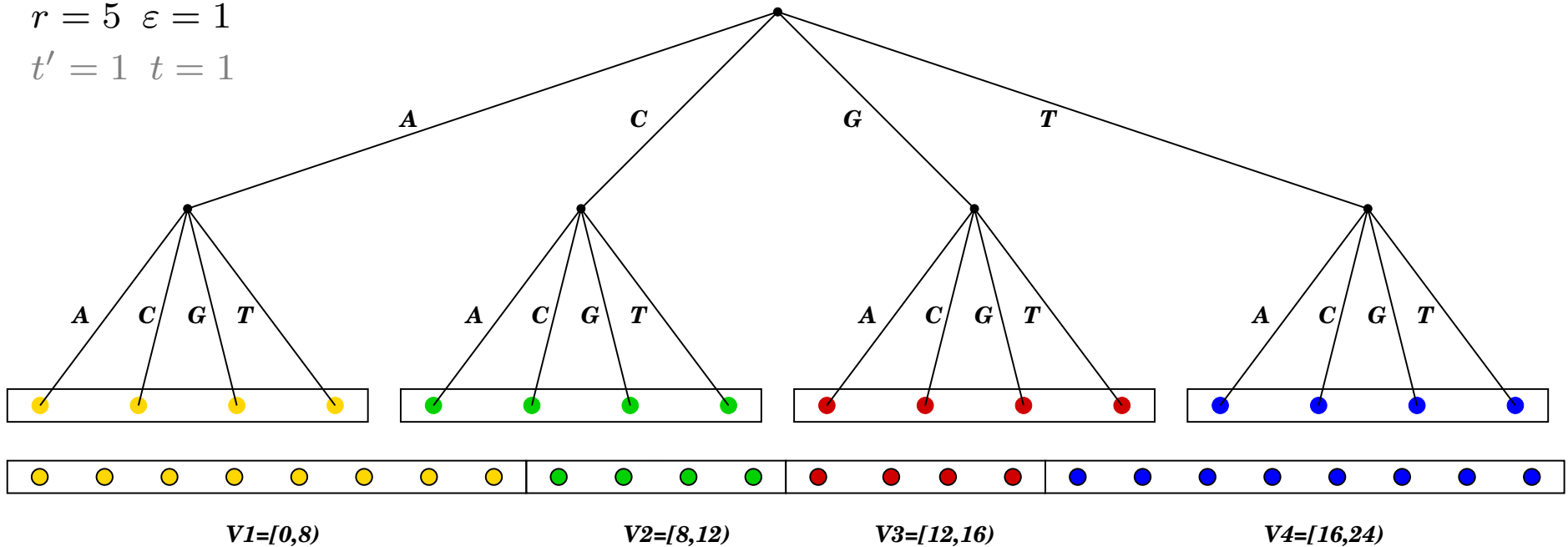
$j$	1	2	3	4
$\sigma_j$	A	C	T	G
$w_j$	2	1	1	2

3. for each  $j \in 1, \dots, \ell$

$$4. \quad V_j = \left[ \sum_{k=1}^{j-1} w_k \times c^t, \sum_{k=1}^j w_k \times c^t \right)$$

$r = 5 \quad \varepsilon = 1$

$t' = 1 \quad t = 1$



# Parallelization

$j$	1	2	3	4
$\sigma_j$	A	C	T	G
$w_j$	2	1	1	2

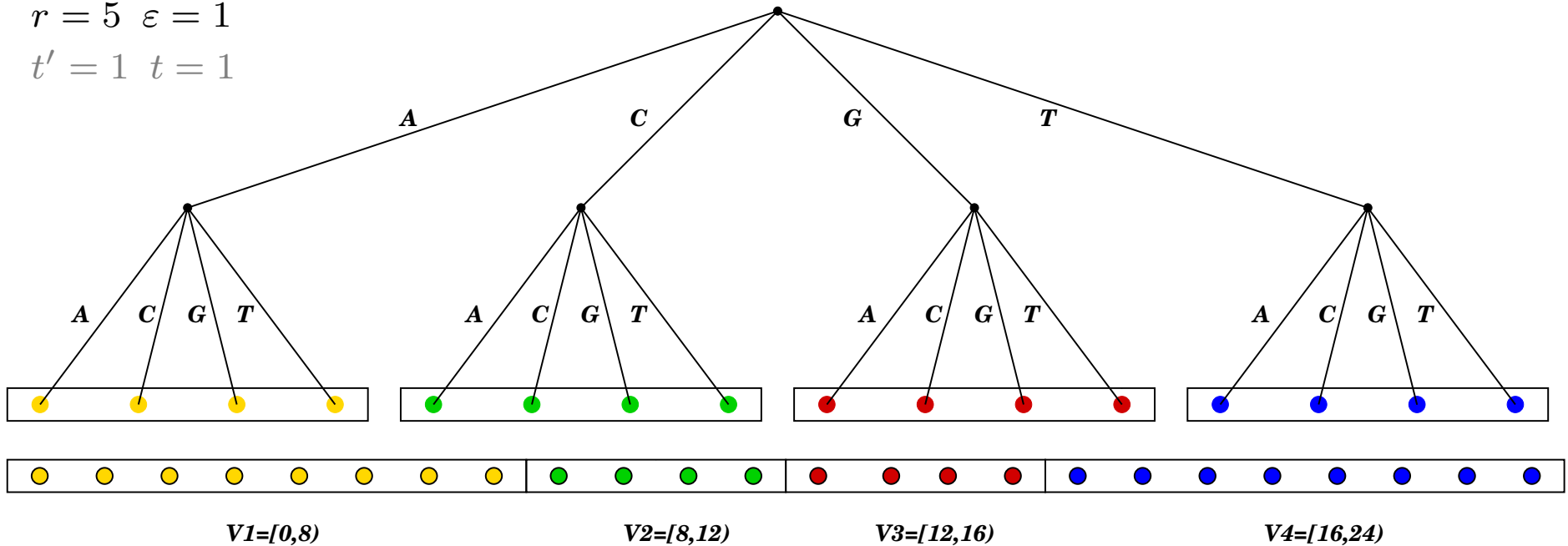
$$5. w = \sum_{j=1}^{\ell} w_j$$

$$6. \gamma = w \times c^t \bmod r$$

$$7. \delta = \left\lfloor \frac{w \times c^t}{r} \right\rfloor$$

$$r = 5 \quad \varepsilon = 1$$

$$t' = 1 \quad t = 1$$





# Parallelization

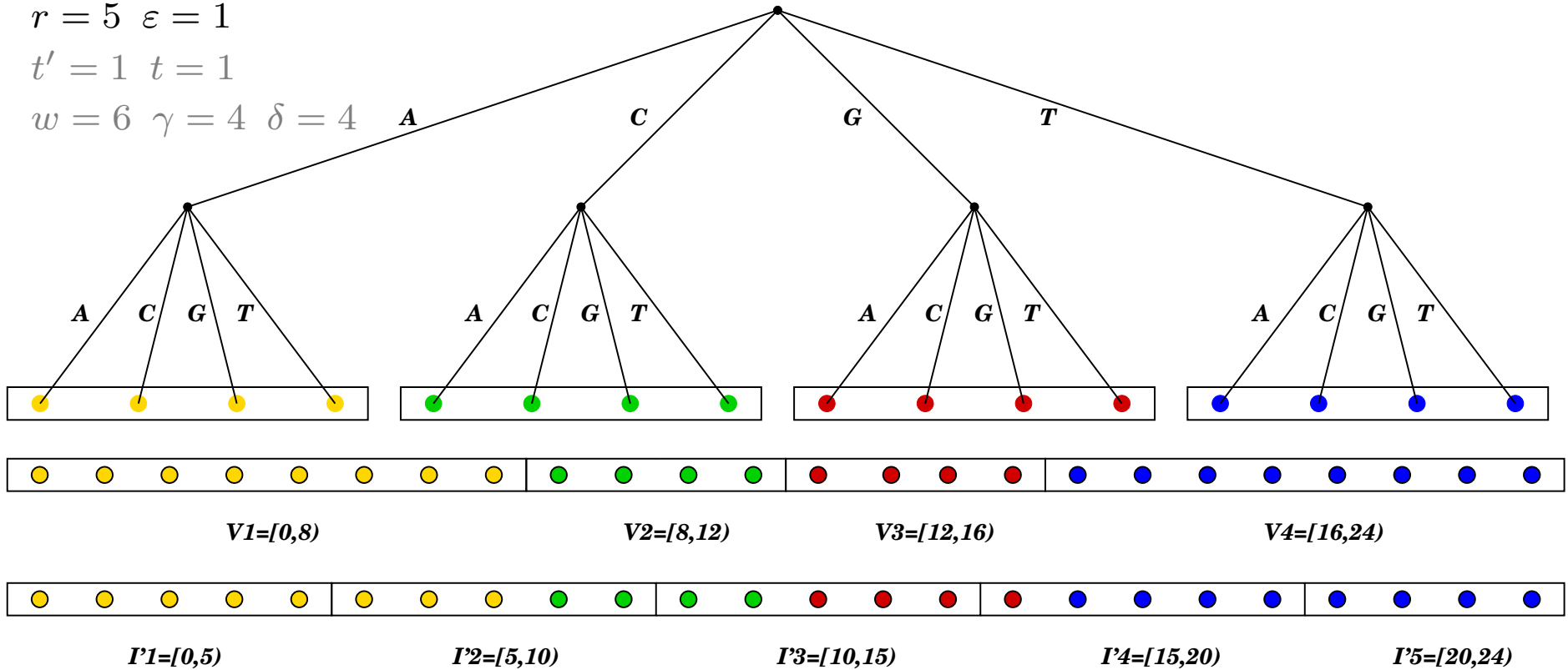
$j$	1	2	3	4
$\sigma_j$	A	C	T	G
$w_j$	2	1	1	2

$$8. I'_i = \begin{cases} [(i-1)(\delta+1), i(\delta+1)) & \text{for all } i \leq \gamma \\ [\gamma(\delta+1) + (i-(\gamma+1))\delta, \gamma(\delta+1) + (i-\gamma)\delta) & \text{otherwise} \end{cases}$$

$r = 5 \quad \varepsilon = 1$

$t' = 1 \quad t = 1$

$w = 6 \quad \gamma = 4 \quad \delta = 4$



# Parallelization

$j$	1	2	3	4
$\sigma_j$	A	C	T	G
$w_j$	2	1	1	2

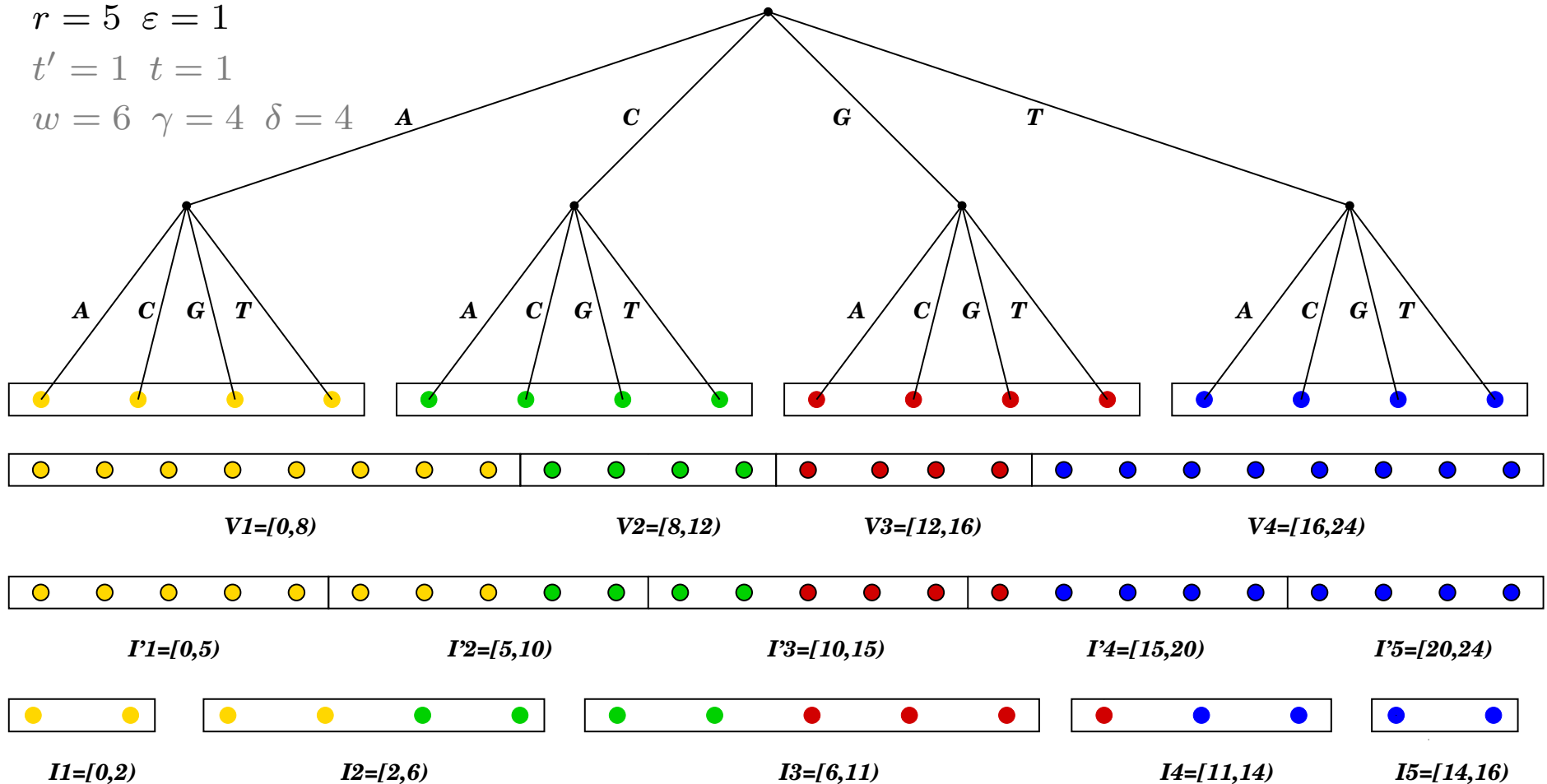
9. transform  $I'_i = [a, b)$  into  $I_i = [f(a), f(b))$  with

$$f(x) = \begin{cases} (j-1) \times c^t + \frac{x - \inf(V_j)}{w_j} & \text{for all } x \in V_j \\ \ell \times c^t & \text{if } x = w \times c^t \end{cases}$$

$r = 5$   $\varepsilon = 1$

$t' = 1$   $t = 1$

$w = 6$   $\gamma = 4$   $\delta = 4$



# Parallelization

**PExtractModels** ( **Model**  $m$ , **Block**  $i$ , **PartitionSet**  $I_i$  of  $\mathcal{M}$  )

1. for each node-occurrence  $v$  of  $m$
2.     if ( $i > 1$ )
3.         put in *PotencialStarts* the children of  $v$  at levels  
             $(i - 1)k + (i - 1)d_{min_{i-1}}$  to  $(i - 1)k + (i - 1)d_{max_{i-1}}$
4.     else
5.         put  $v$  in *PotencialStarts*
6. for each model  $m_i \in I_i$  obtained by doing a recursive depth-first traversal from the root of the virtual model tree  $\mathcal{M}$  while simultaneously traversing  $\mathcal{T}$  from the node-occurrences in *PotencialStarts*
7.     if ( $i < p$ )
8.         **PExtractModels** ( $m = m_1 \dots m_i, i + 1, I_i$ )
9.     else
10.         **KeepModel** ( $\langle (m_1, \dots, m_p), ((d_{min_1}, d_{max_1}), \dots, (d_{min_p}, d_{max_p})) \rangle$ )

# Parallelization

**PSmile** ( GridNode  $i$ , WorkOverload  $\varepsilon$  )

1. compute weights  $(w_i)_{1 \leq i \leq |\Sigma|}$ ;
2. build suffix tree  $\mathcal{T}$ ;
3. create colors on  $\mathcal{T}$ ;
4. let  $I_i = \text{SimpleCut}(i, |\Sigma|, r, (w_i)_{1 \leq i \leq |\Sigma|}, |\Sigma|, \varepsilon)$ ;
5. call  $\text{PExtractModels}(\mathcal{T}, I_i)$ ;

**Proposition.** Assume  $\Sigma$  fixed and  $w_i = 1$  for  $1 \leq i \leq |\Sigma|$ . The parallel algorithm PSmile is work-efficient with respect to the sequential version when  $r = O(\nu^{\frac{p}{2}}(e, k))$  and  $\frac{\varepsilon}{w} \leq \frac{1}{r}$ .

# Parallelization

## Experimental results

	2 boxes		3 boxes	
	models	time (sec)	models	time (sec)
grid node 1	2	155.83	9987	444.50
grid node 2	1	168.11	6178	385.28
grid node 3	2	245.35	3108	473.70
grid node 4	16	262.51	15884	581.64
total	21	831.80	35157	1885.18
parallel time	262.51		581.64	
sequential time	757.97		1790.70	
speed up	2.9		3.1	

# On going and future work

- Implementation of a more efficient sequential algorithm to extract structured models  
[L. Marsan and M.-F. Sagot, J. Computational Biology, 2000]
- Establishing an even more efficient algorithm to extract structured models  
[A. Carvalho, A. Freitas, A. Oliveira and M.-F. Sagot, in preparation, 2003]
- Comparison between algorithms which attempts to model the combinatorics of regulation