A parallel algorithm for the extraction of structured motifs

Alexandra Carvalho

joint work with

A. Freitas, A. Oliveira, M.-F. Sagot

ALGOS, INESC-ID

Plan of the talk

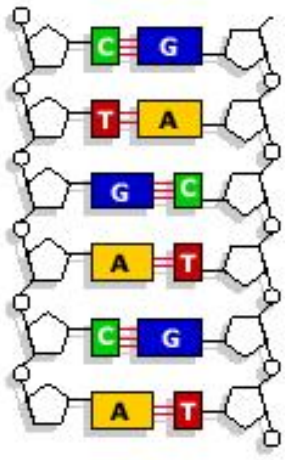
Biological motivation:

- Promoter and Regulatory Sequences

Computational approach:

- 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, ACM SAC BIO, 2004]**
  - The PARTITION UP TO ε problem
    - The SimpleCut algorithm
  - The tree partition problem
    - The PSMILE algorithm
  - Experimental results

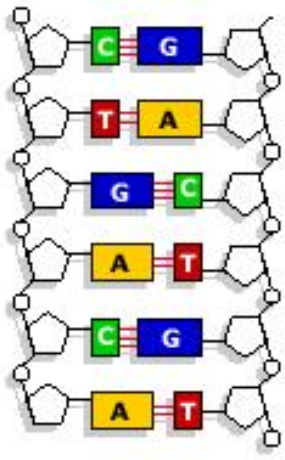
Promoter and Regulatory Sequences



DNA - DeoxyriboNucleic Acid:

- contain the bases A, C, G, and T
- double-stranded molecule

Promoter and Regulatory Sequences



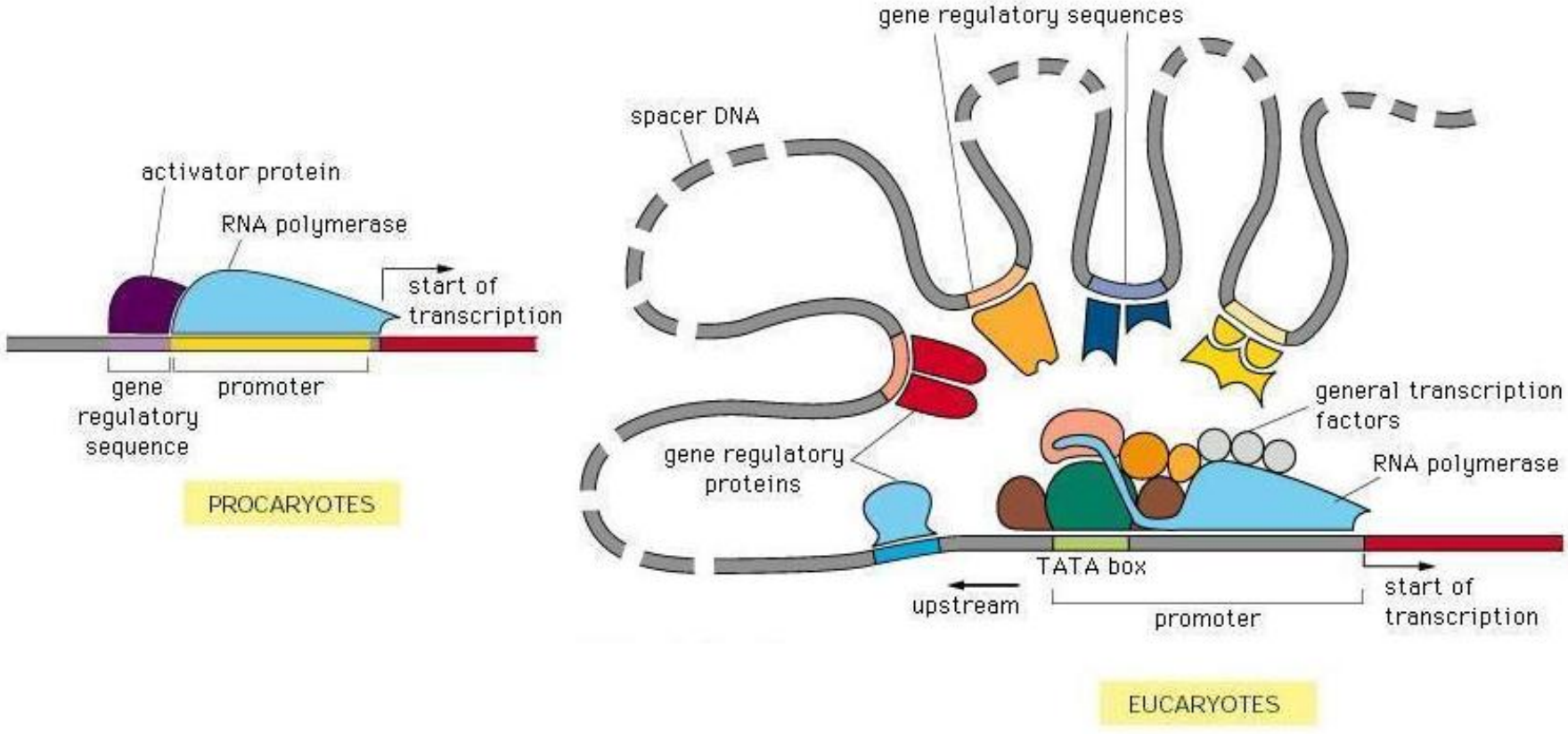
DNA - DeoxyriboNucleic Acid:

- contain the bases A, C, G, and T
- double-stranded molecule

Classification of living organisms:

- Prokaryotes:
  - Greek words: *pro* $\equiv$ "before"and *karyon* $\equiv$ "nucleus"
  - bacteria and prokaryotes are generally used interchangeably
- Eukaryotes:
  - Greek words: *eu* $\equiv$ "well"and *karyon* $\equiv$ "nucleus"

Promoter and Regulatory Sequences



Structured motifs

Definition. ***model***

A model is an element in Σ^+.

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

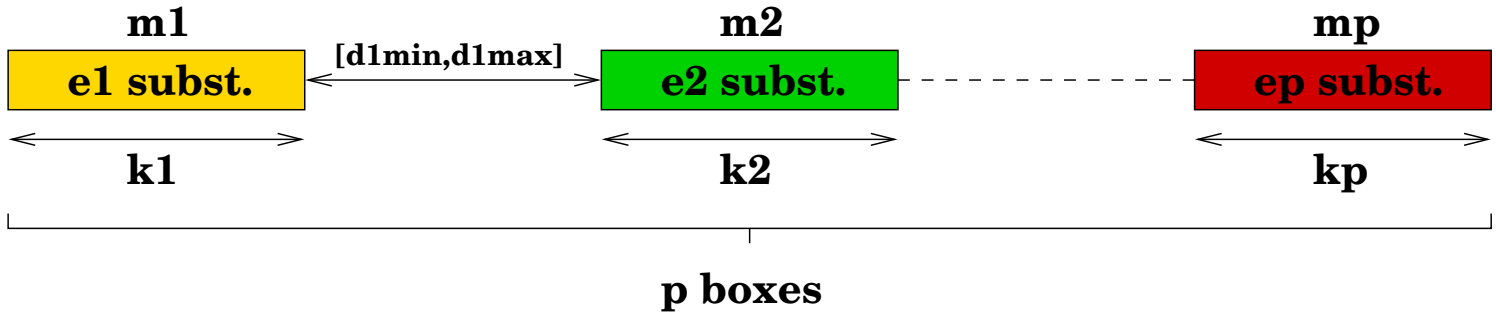
Definition. ***model***

A model is an element in Σ^+.

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 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 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.

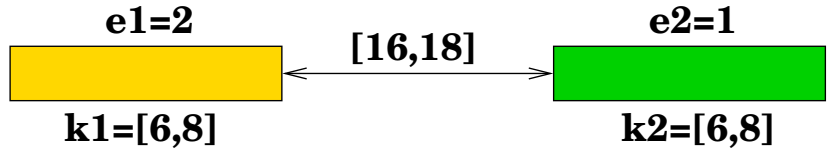
Structured motifs

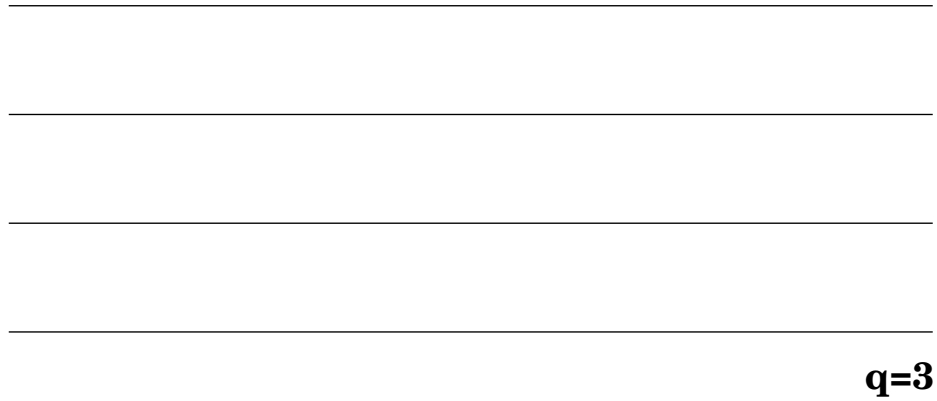
Definition. ***e-occurrence***

A model m e-occurs in the input sequences if exists u in the input sequences such that 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.





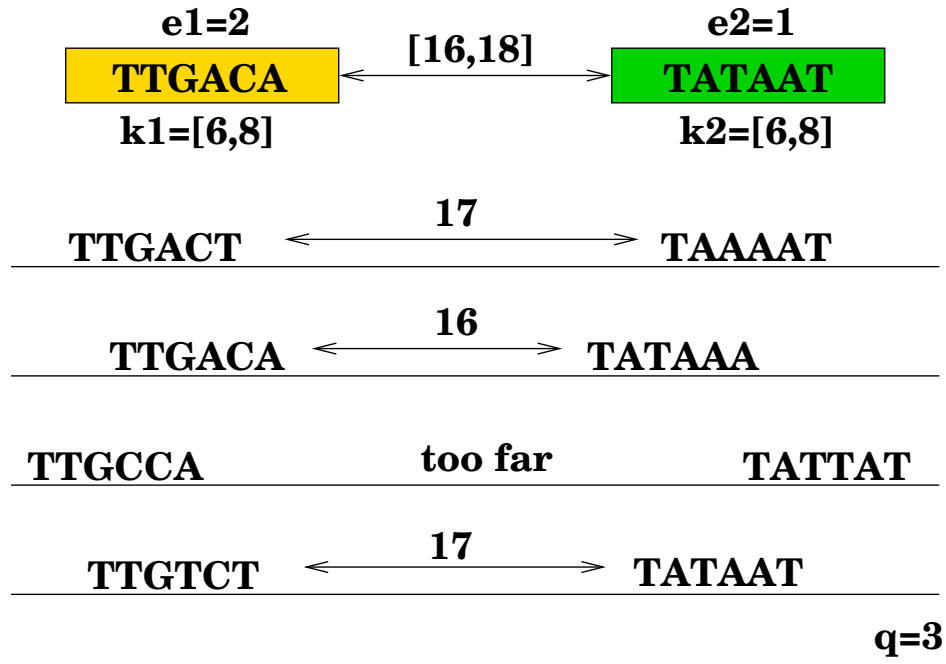
Structured motifs

Definition. ***e-occurrence***

A model m e-occurs in the input sequences if exists u in the input sequences such that 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
CCGTTTCAGTTATAGTTAATATGTAGCCTTTTTAGGCAATGAAAAAACTTTGAAA

>strand - yaaI similar to isochorismatase
TTTCAAAGTTTTTTCATTGCCTAAAAAGGCTACATATTAACTATAACTGAAACGG

>strand + metS methionyl-tRNA synthetase
ATTTTATAAATATTTAATAAAGCTATTATCCTACTAAAAATCCTTTTAAATCAAGACTTTCGAACCAA
AGTTTTTTATTTCATTTGATTATATACGACAAAATTCGACACGAACAGACTTTTTTTATTTTCATTAA
AGATTTTTAATTTTAATTATTCTTTTTCAGGGCGTATGTATATATTCTTGATCTTAAAGGCTAAGATG
GTATCATAGATAAAGGATAAATATAAATAATATTCATATATGATTTGCACTTATCGCCGCTCTCGTCC
TTTGGGCGGGAGCTTTTTGACATTCTGA
```

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

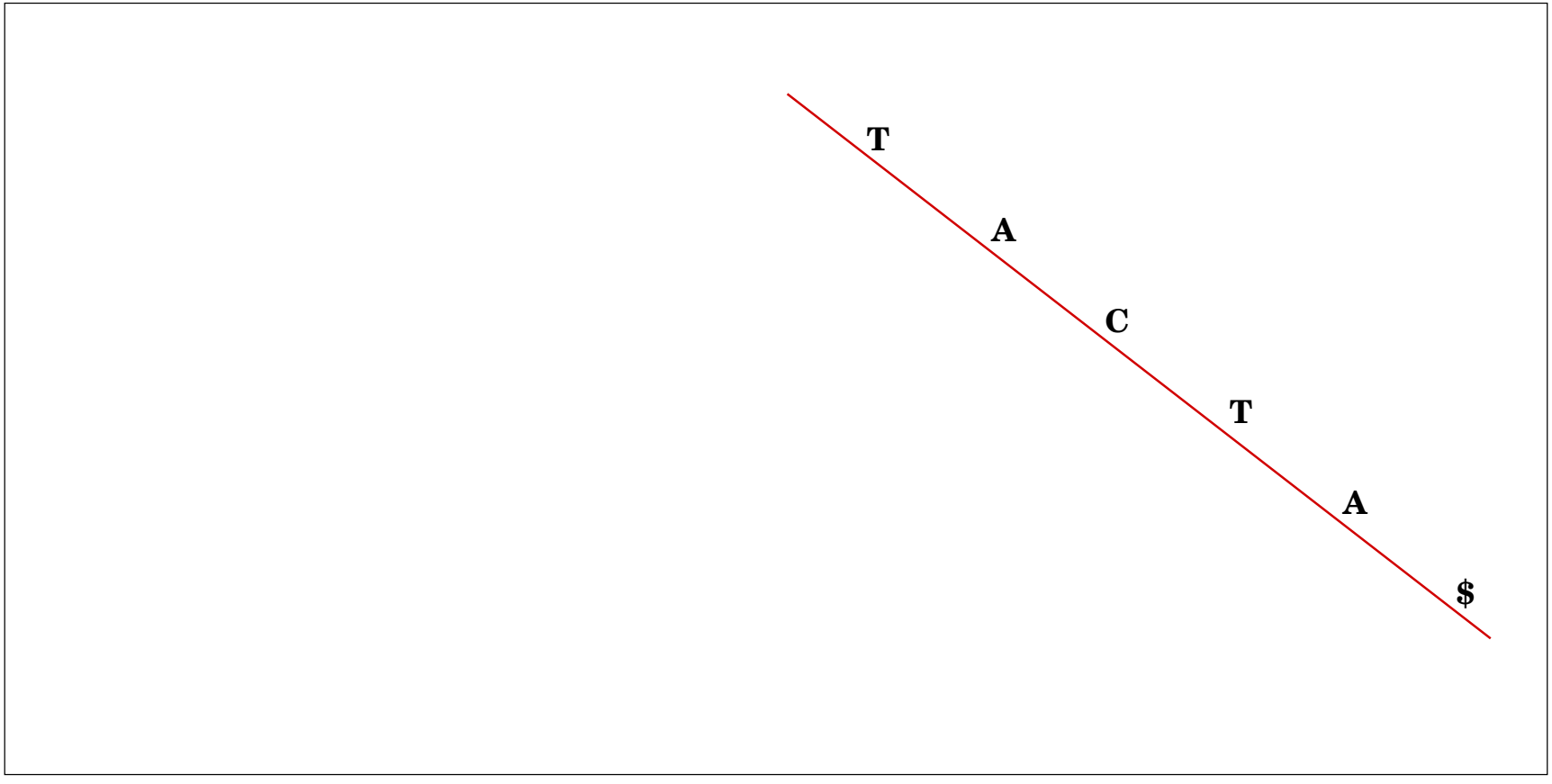
Theorem. Suffix trees can be built in linear-time.

Suffix Tree

Suffix tree for the string TACTA$

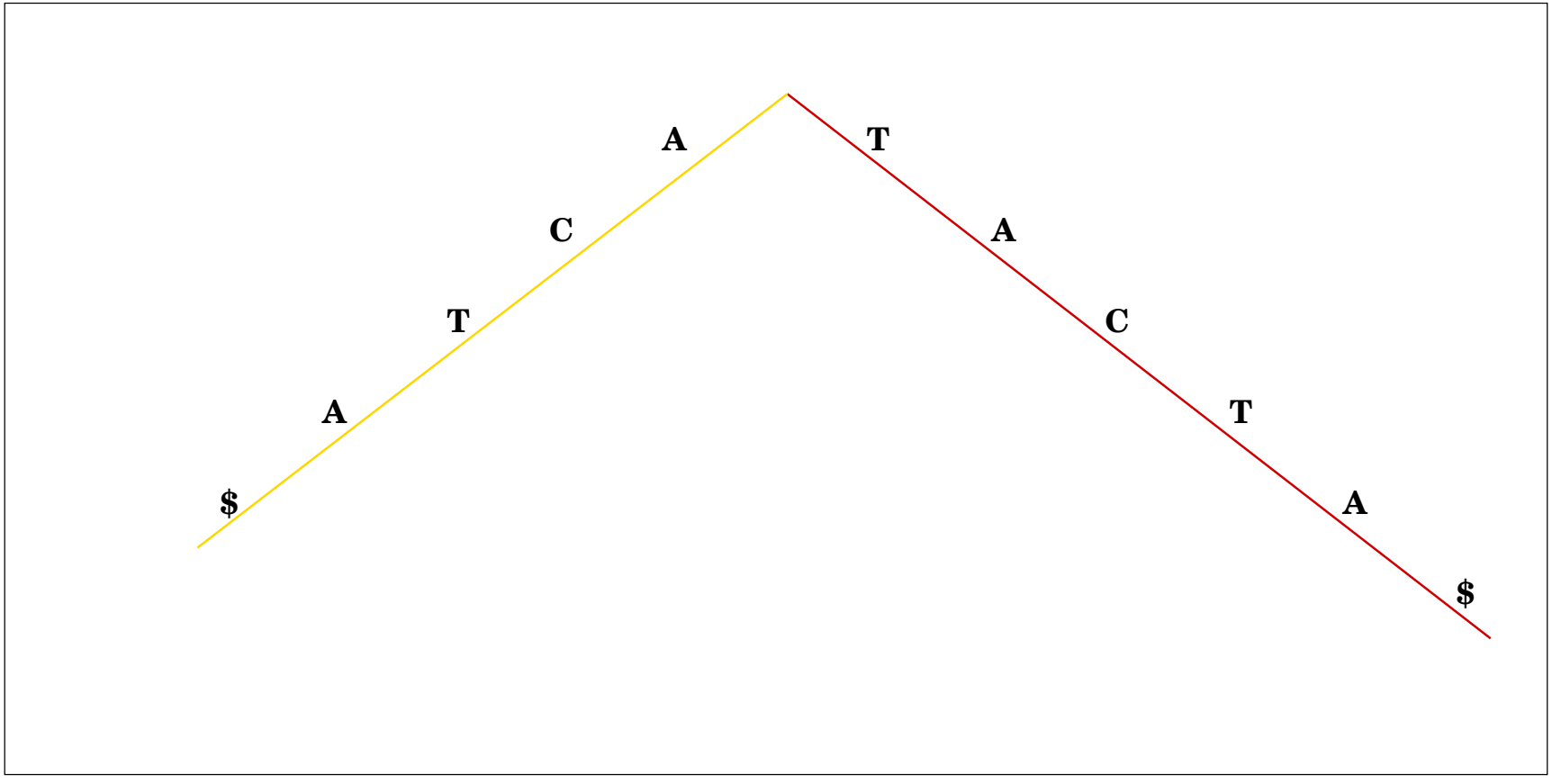
Suffix Tree

Suffix tree for the string TACTA$



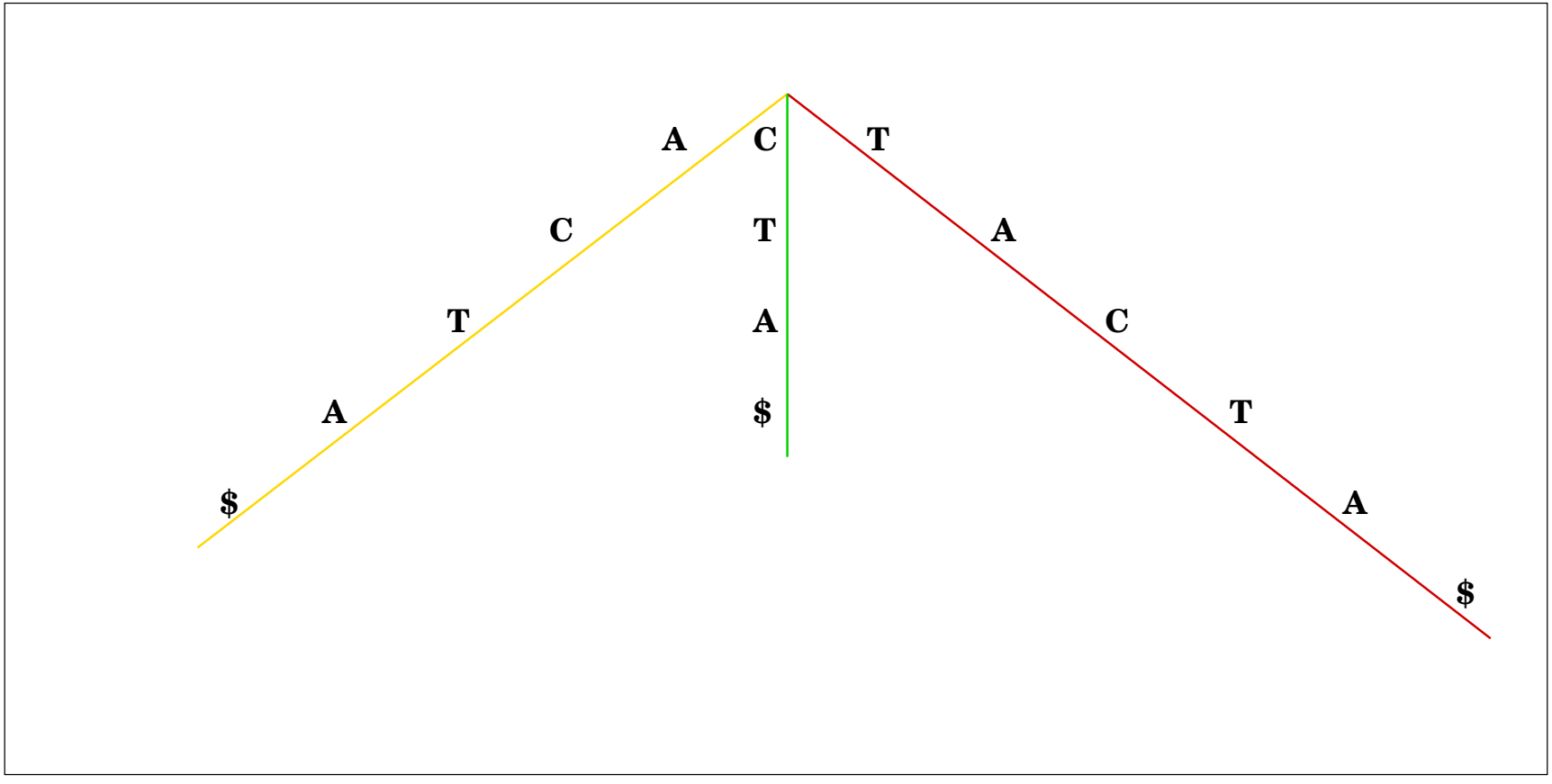
Suffix Tree

Suffix tree for the string TACTA$



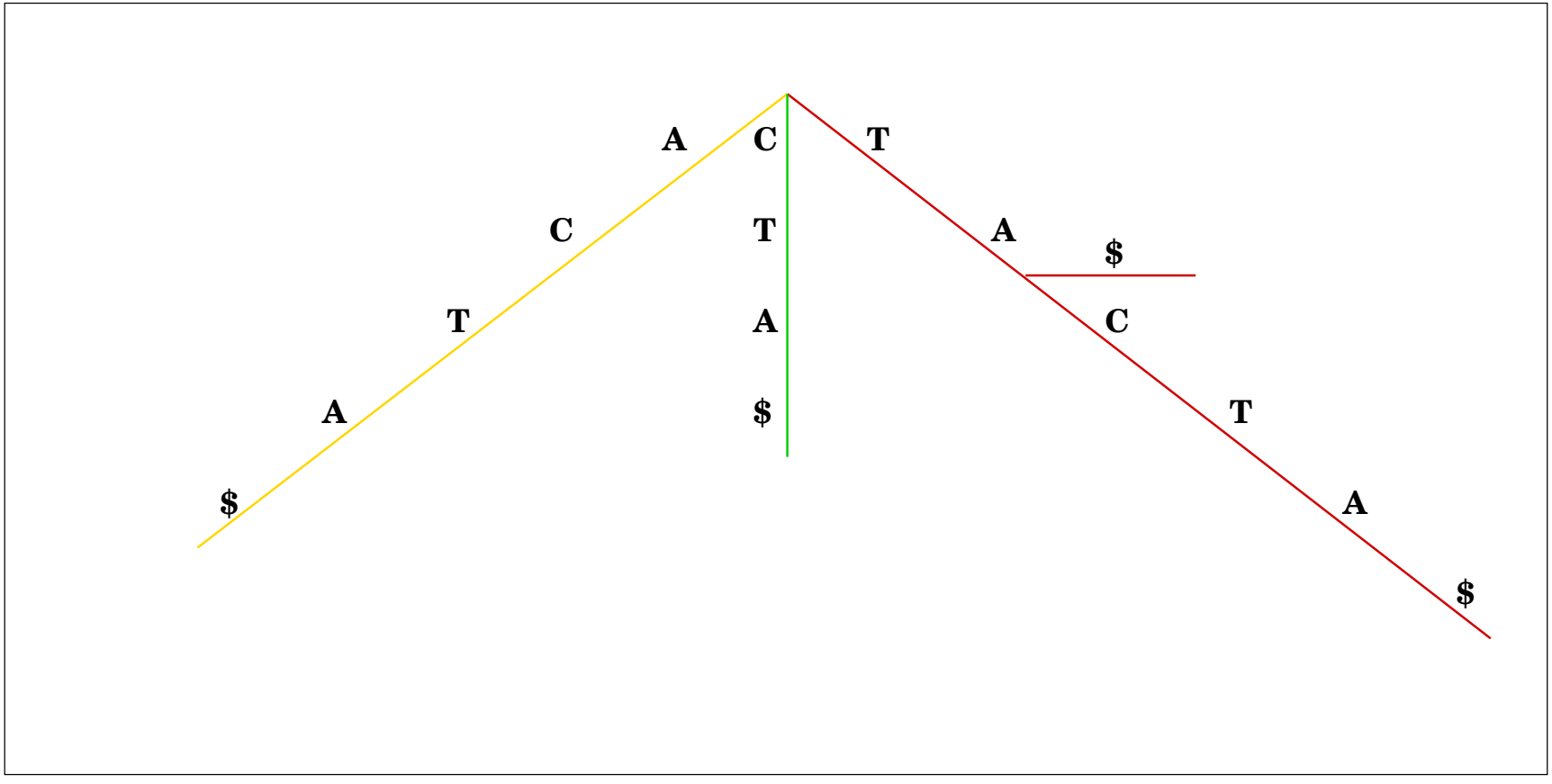
Suffix Tree

Suffix tree for the string TACTA$



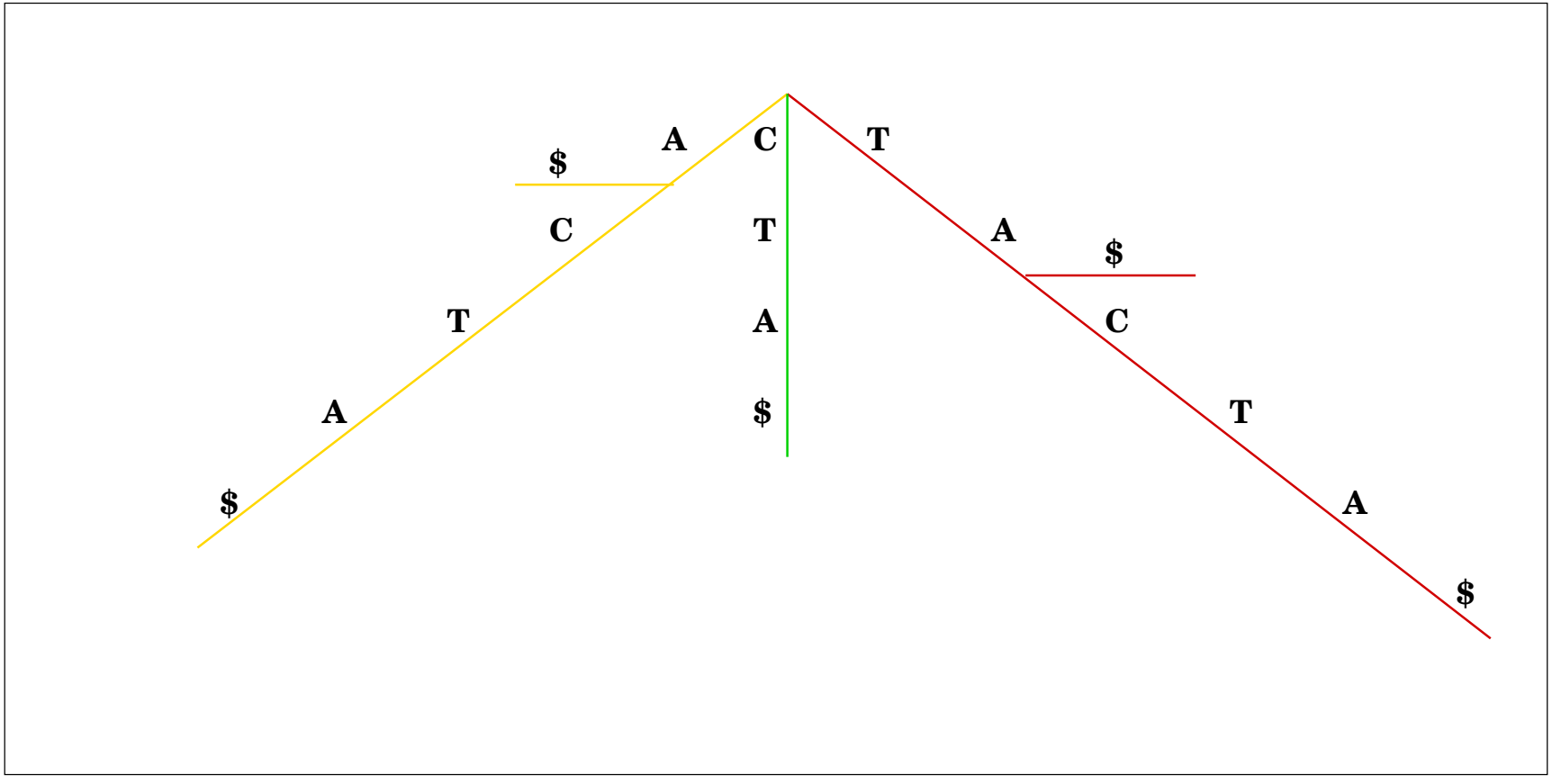
Suffix Tree

Suffix tree for the string TACTA$



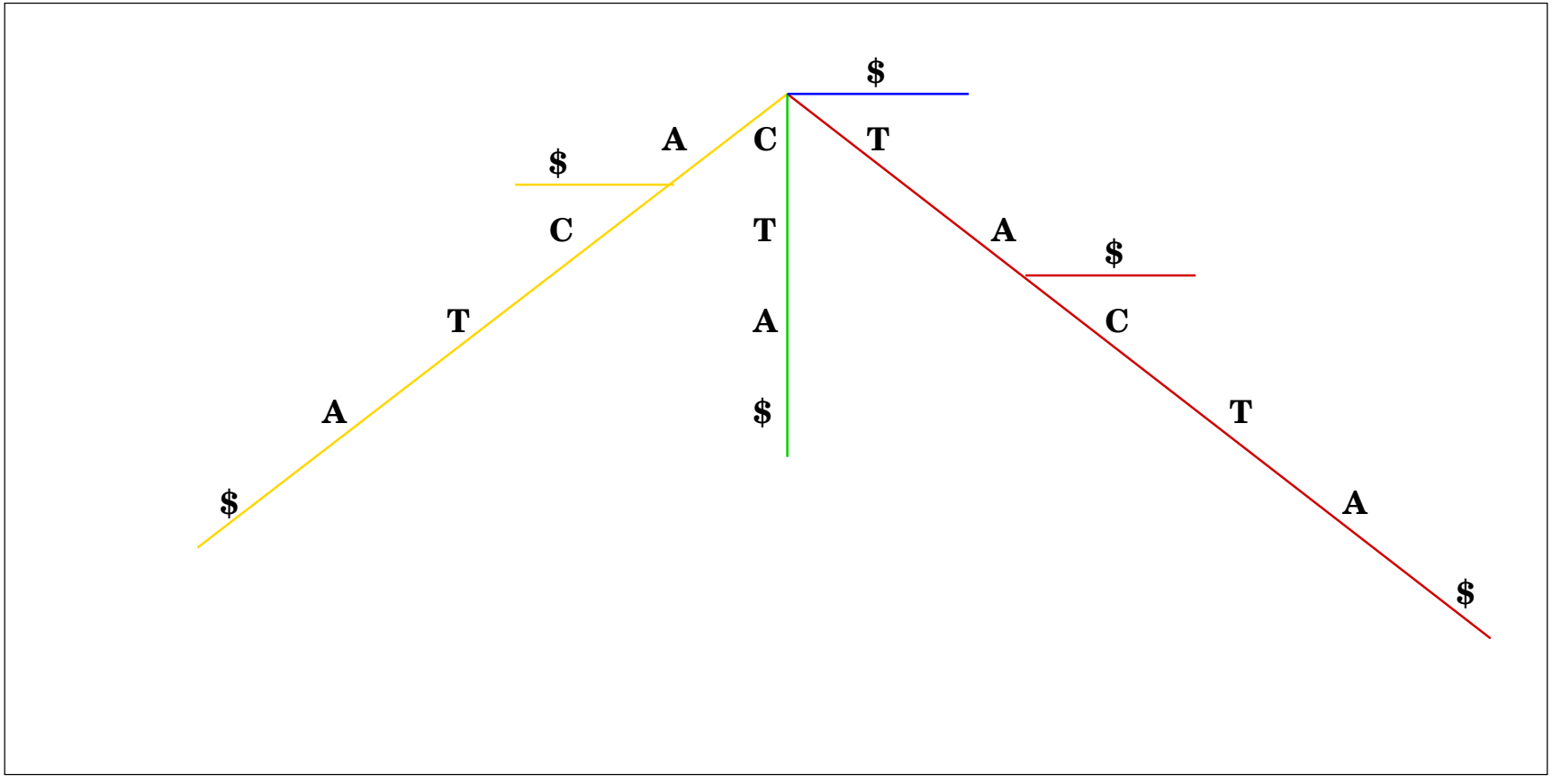
Suffix Tree

Suffix tree for the string TACTA$



Suffix Tree

Suffix tree for the string TACTA$

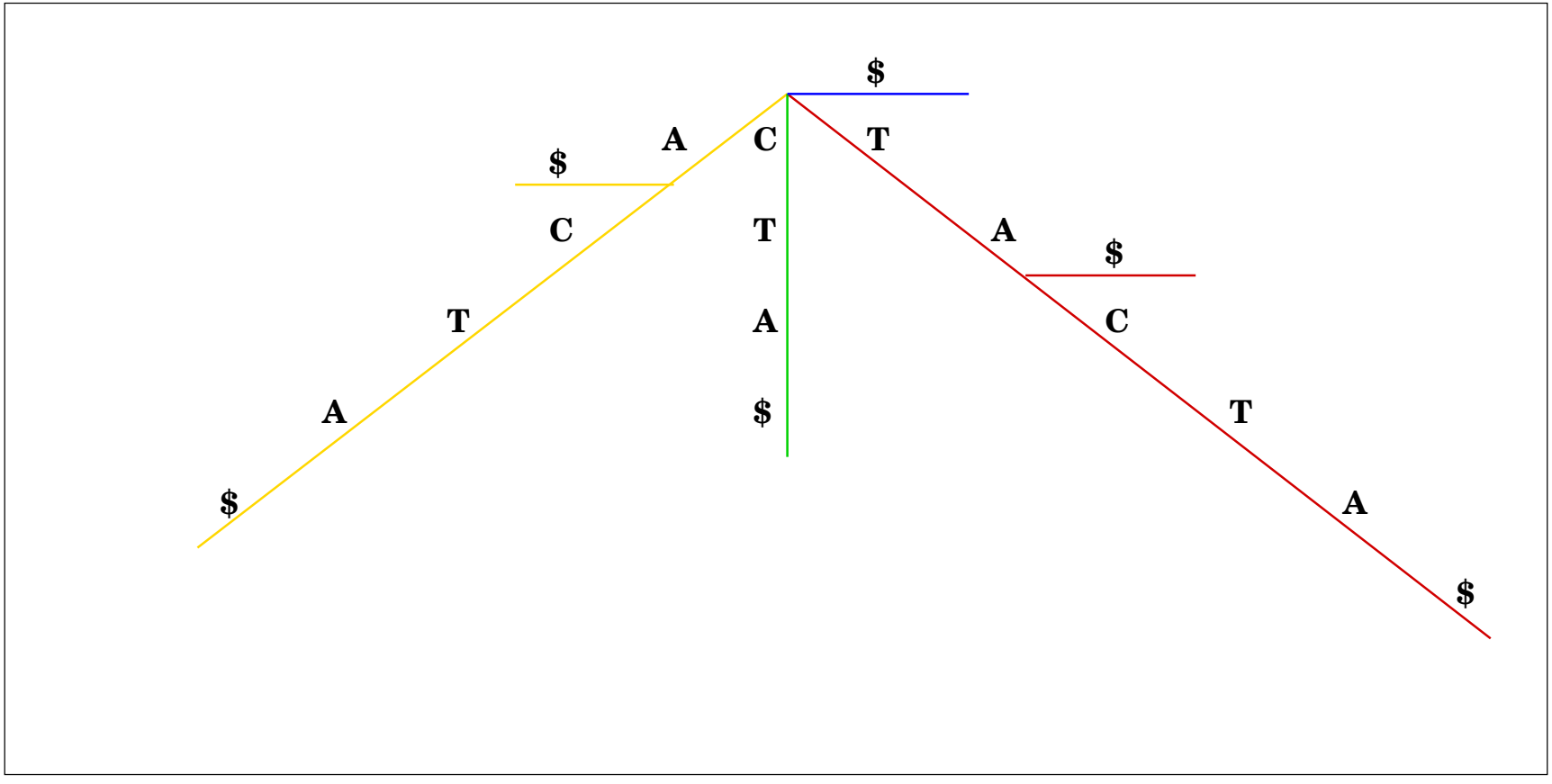


Generalized Suffix Tree

Suffix tree for the strings TACTA$ and CACTCA#

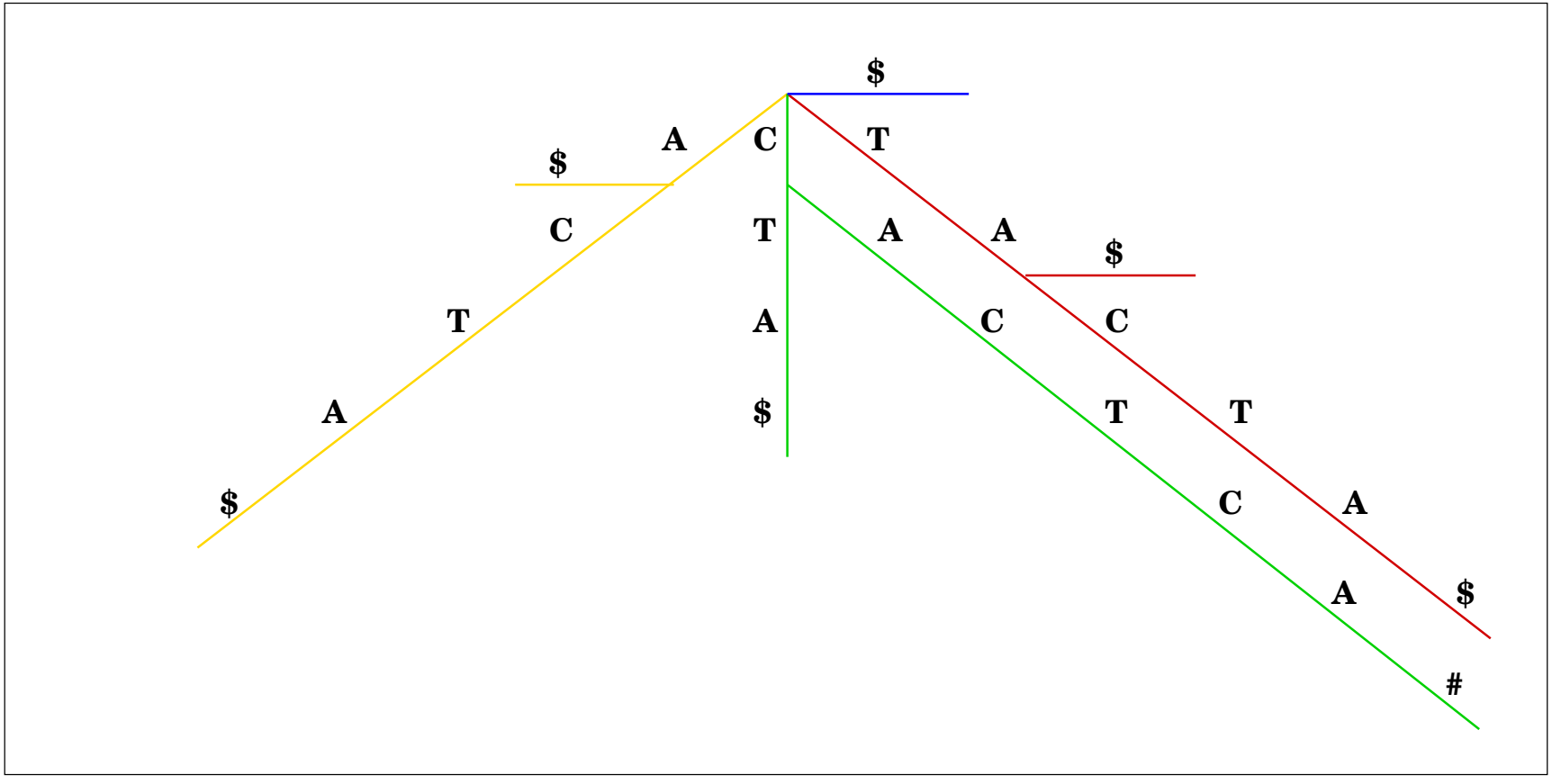
Generalized Suffix Tree

Suffix tree for the strings TACTA$ and CACTCA#



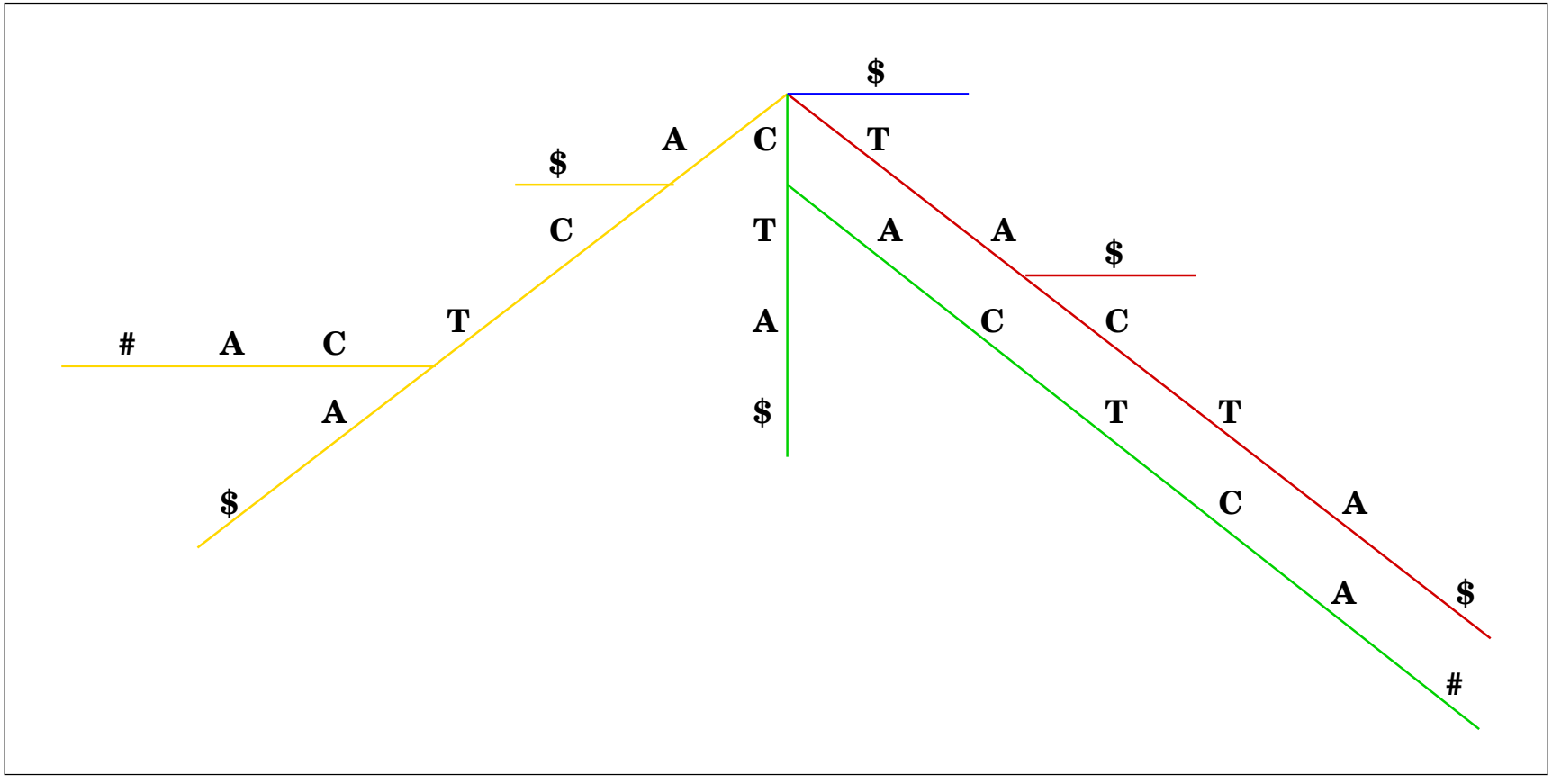
Generalized Suffix Tree

Suffix tree for the strings TACTA$ and CACTCA#



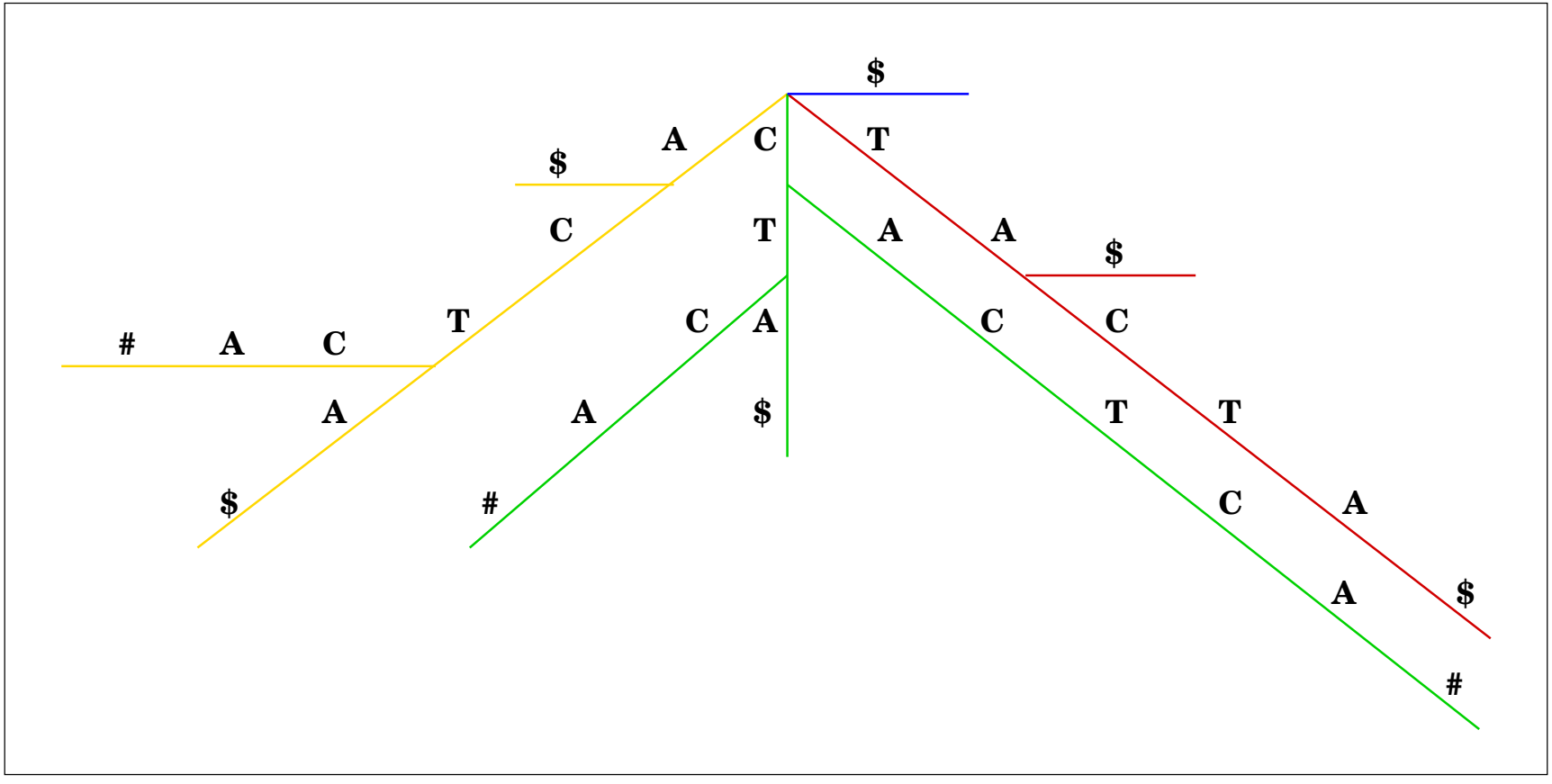
Generalized Suffix Tree

Suffix tree for the strings TACTA$ and CACTCA#



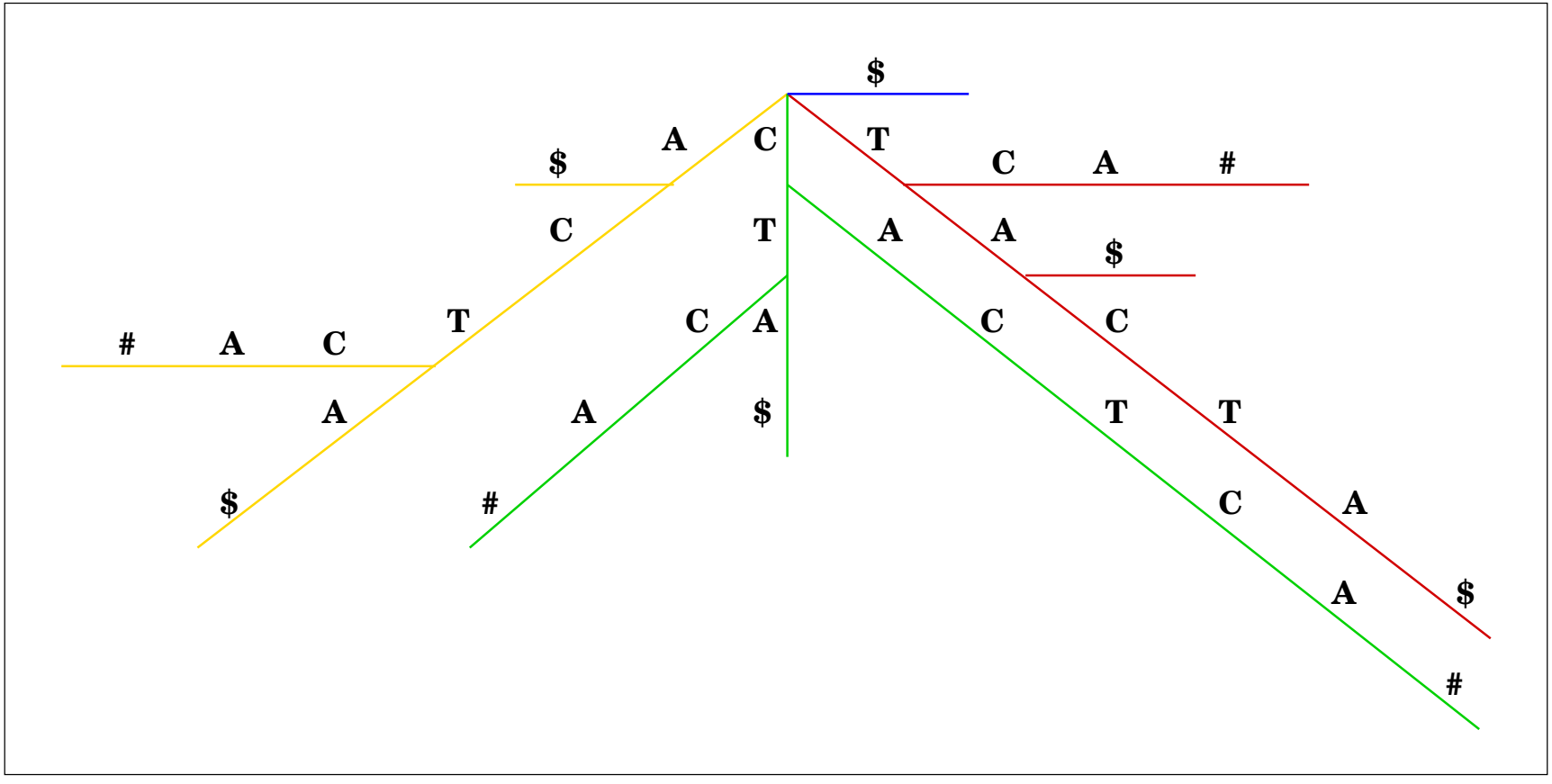
Generalized Suffix Tree

Suffix tree for the strings TACTA$ and CACTCA#



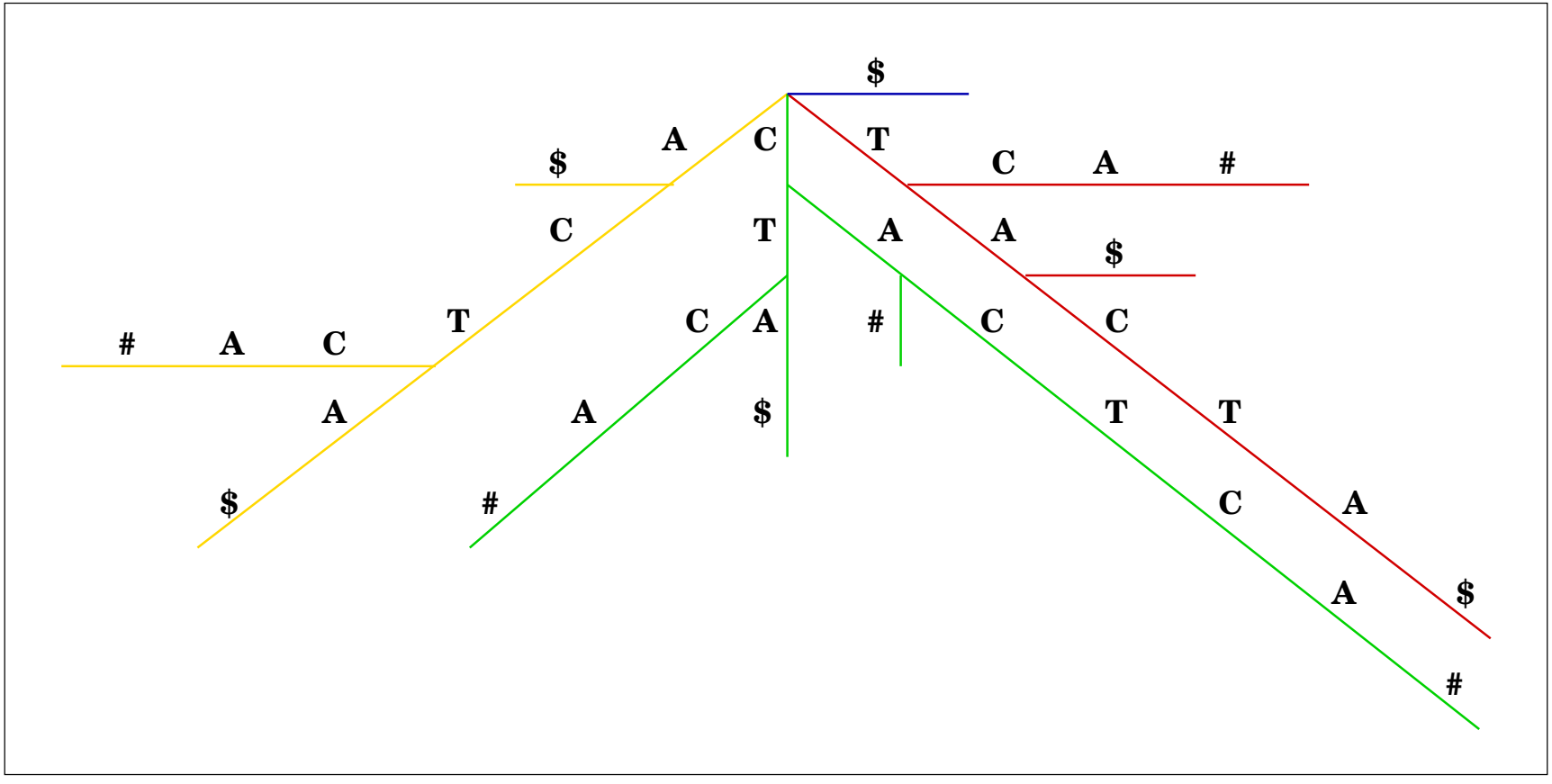
Generalized Suffix Tree

Suffix tree for the strings TACTA$ and CACTCA#



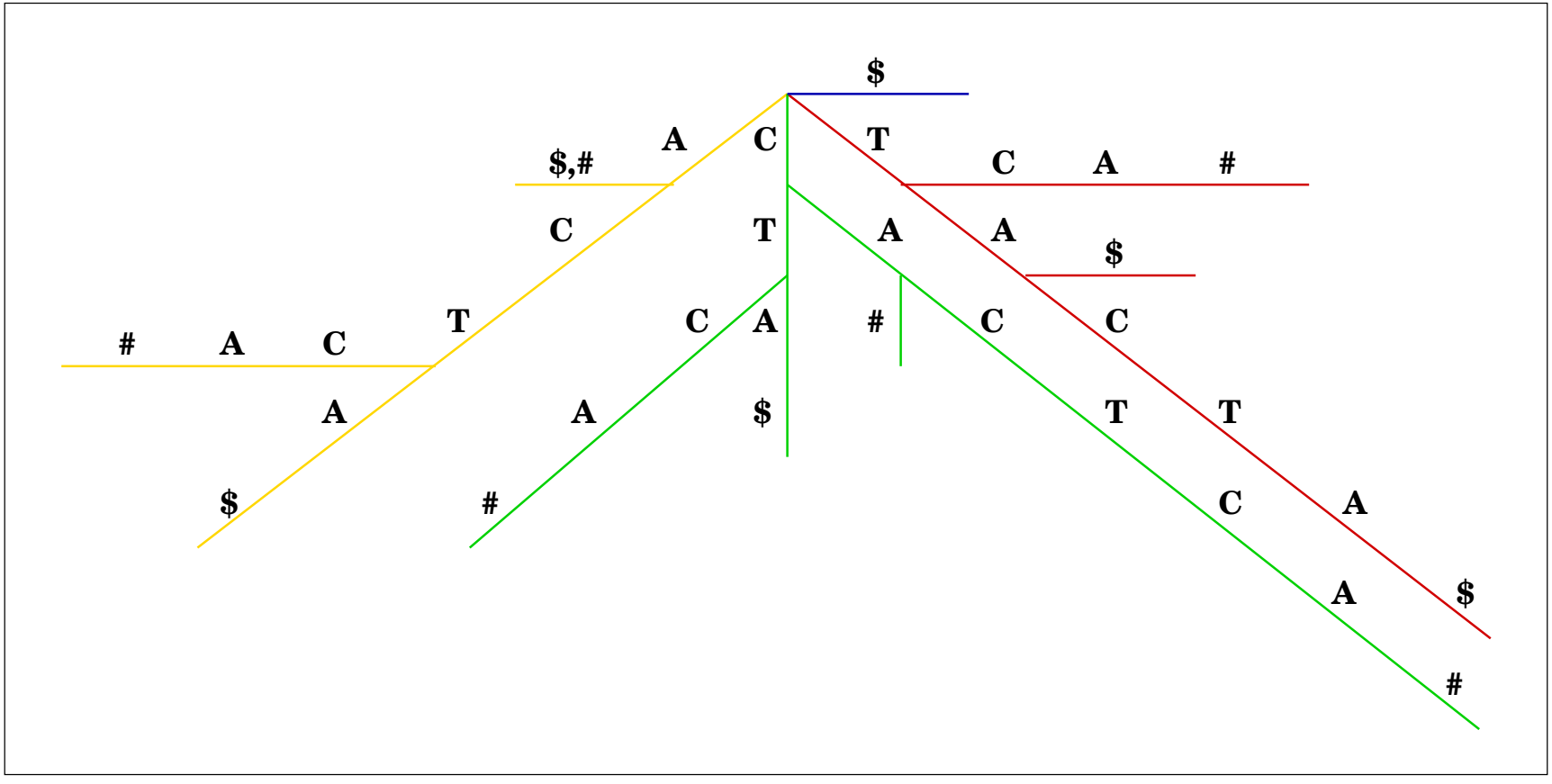
Generalized Suffix Tree

Suffix tree for the strings TACTA$ and CACTCA#



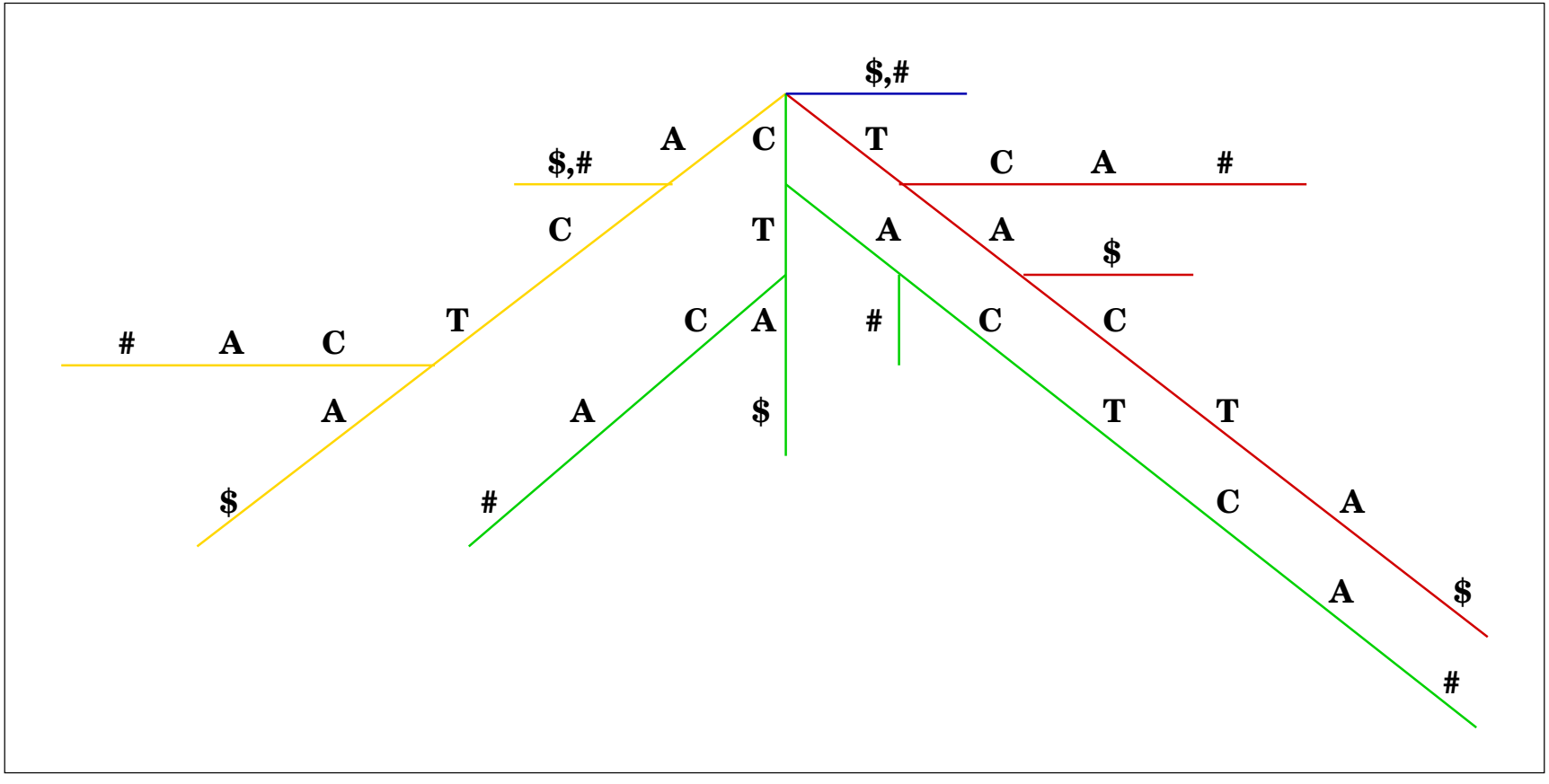
Generalized Suffix Tree

Suffix tree for the strings TACTA$ and CACTCA#



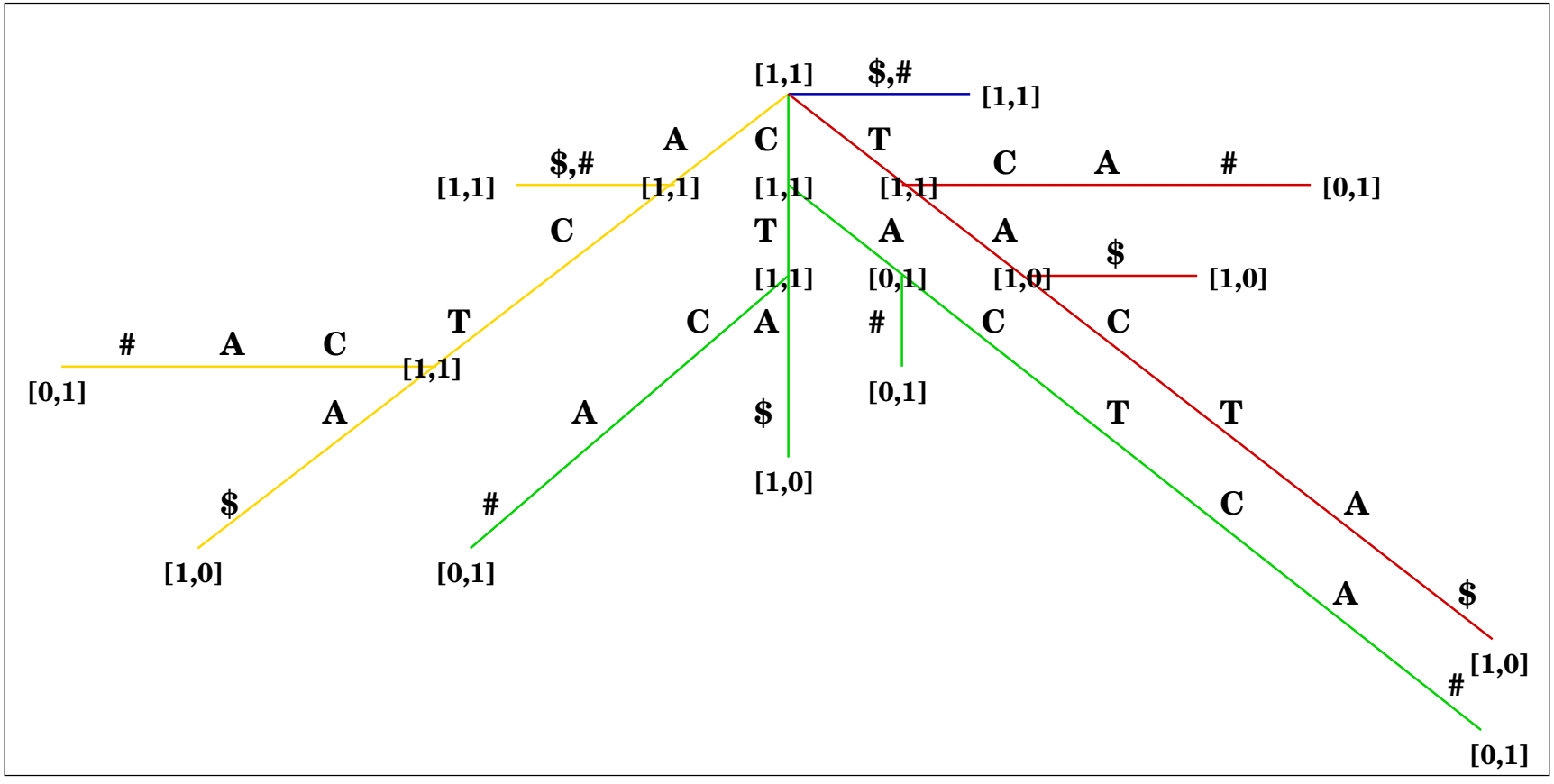
Generalized Suffix Tree

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
- $e_v \leq e$ 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:

$$\nu(e,k) = \sum_{i=0}^{e} \left(\begin{array}{c} k \\ i \end{array} \right) (|\Sigma| - 1)^i \leq k^e |\Sigma|^e.$$

Notation. n_k The number of tree nodes at depth k of a suffix tree.

Extraction of Single Models

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

SpellModels (**Depth** l, **Model** m, **Occurrences** Occ_m)

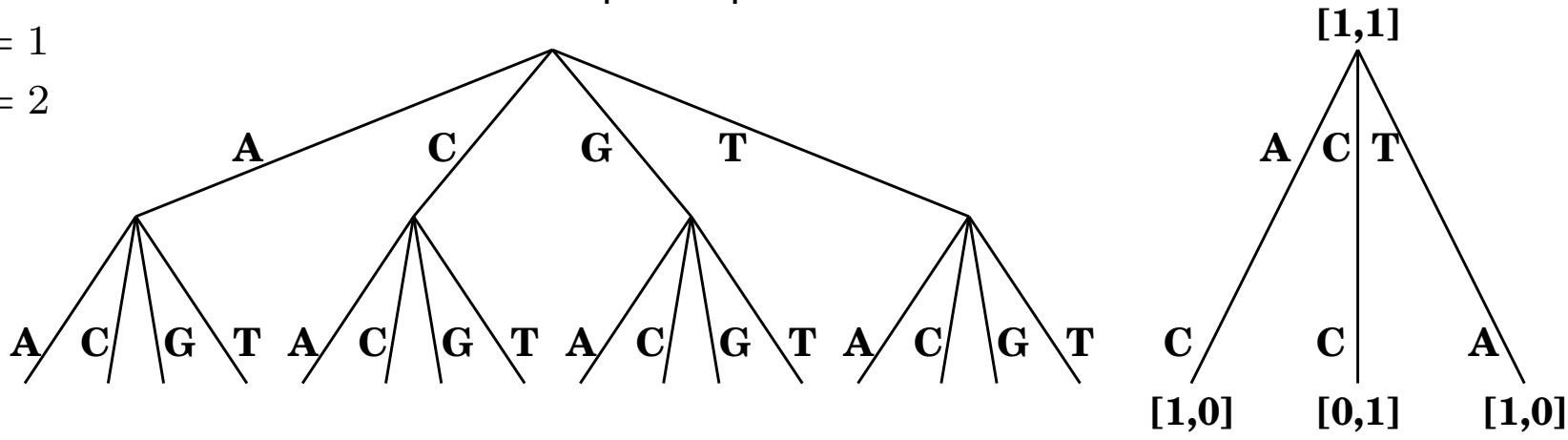
```
 1. if (l==k)
 2.    KeepModel(m)
 3. else
 4.    for each possible symbol α
 5.       Occ_mα = ∅
 6.       for each pair (x, x_err) in Occ_m
 7.          if there is a branch b leaving node x
                with a label starting with α
 8.             x' = node reached by following branch b from x
 9.             add to Occ_mα the pair (x', x_err)
10.          if (x_err < e)
11.             for each branch b leaving x
                    except the one labeled with α
12.                x' = node reached by following branch b from x
13.                add to Occ_mα the pair (x', x_err + 1)
14.       if (there is quorum q)
15.          SpellModels(l + 1, mα, Occ_mα)
```

Extraction of Single Models

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

$k = 2$
$e = 1$
$q = 2$

Input sequences: TAC and CCC

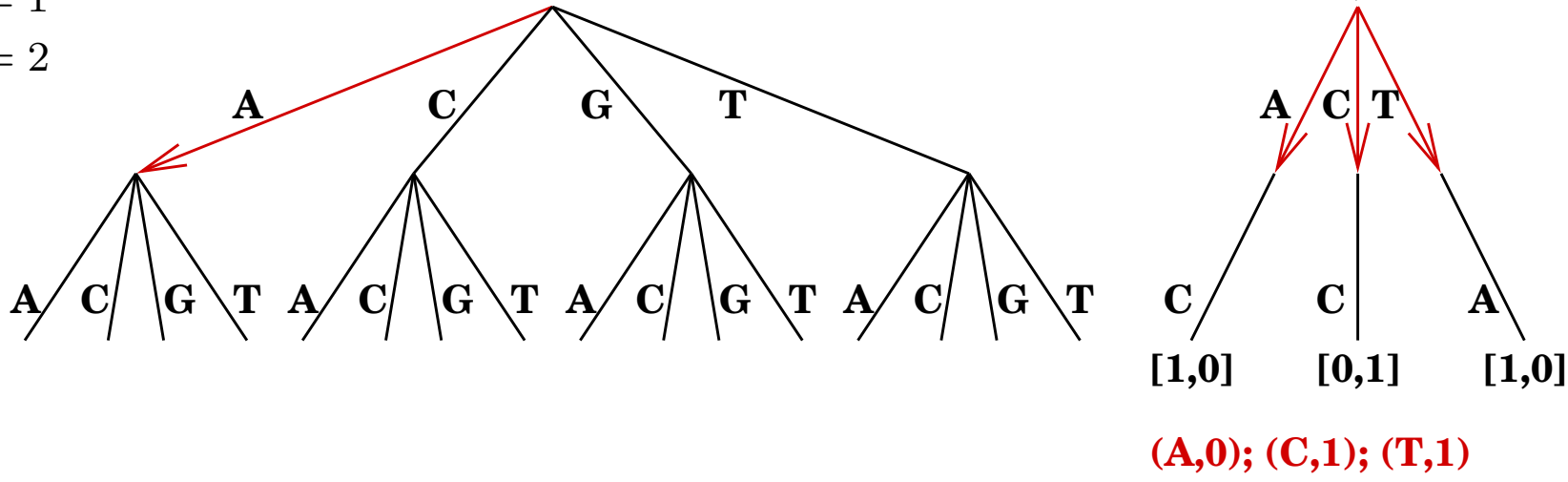


Extraction of Single Models

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

$k = 2$
$e = 1$
$q = 2$

Input sequences: TAC and CCC

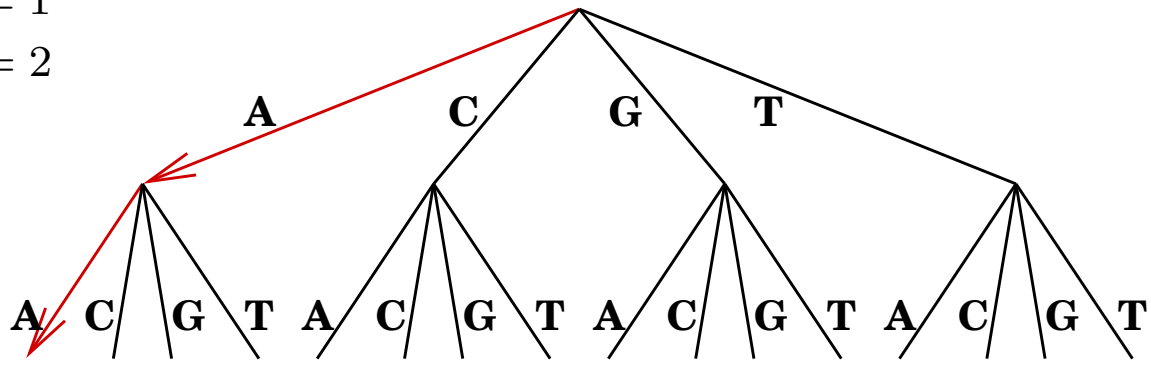


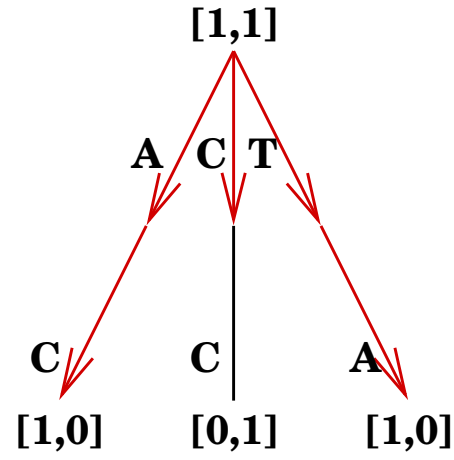
Extraction of Single Models

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

$k = 2$
$e = 1$
$q = 2$

Input sequences: TAC and CCC





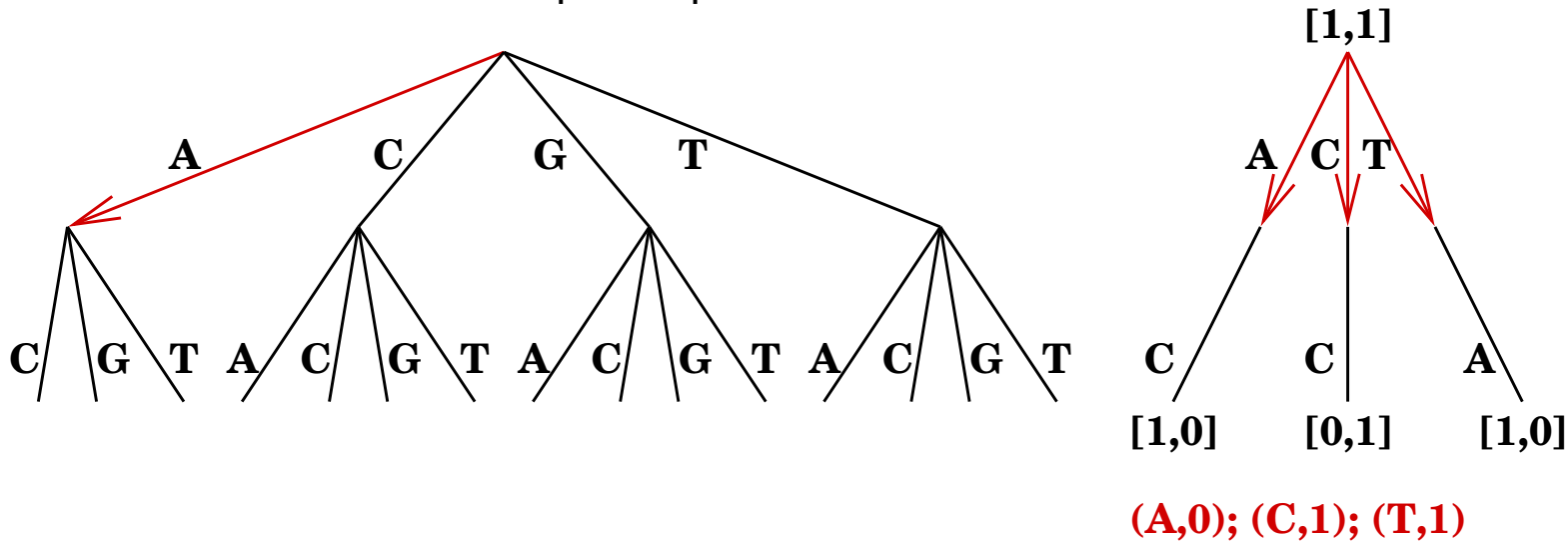
(A,0); (C,1); (T,1)
(AC,1); (TA,1)

Extraction of Single Models

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

$k = 2$
$e = 1$
$q = 2$

Input sequences: TAC and CCC

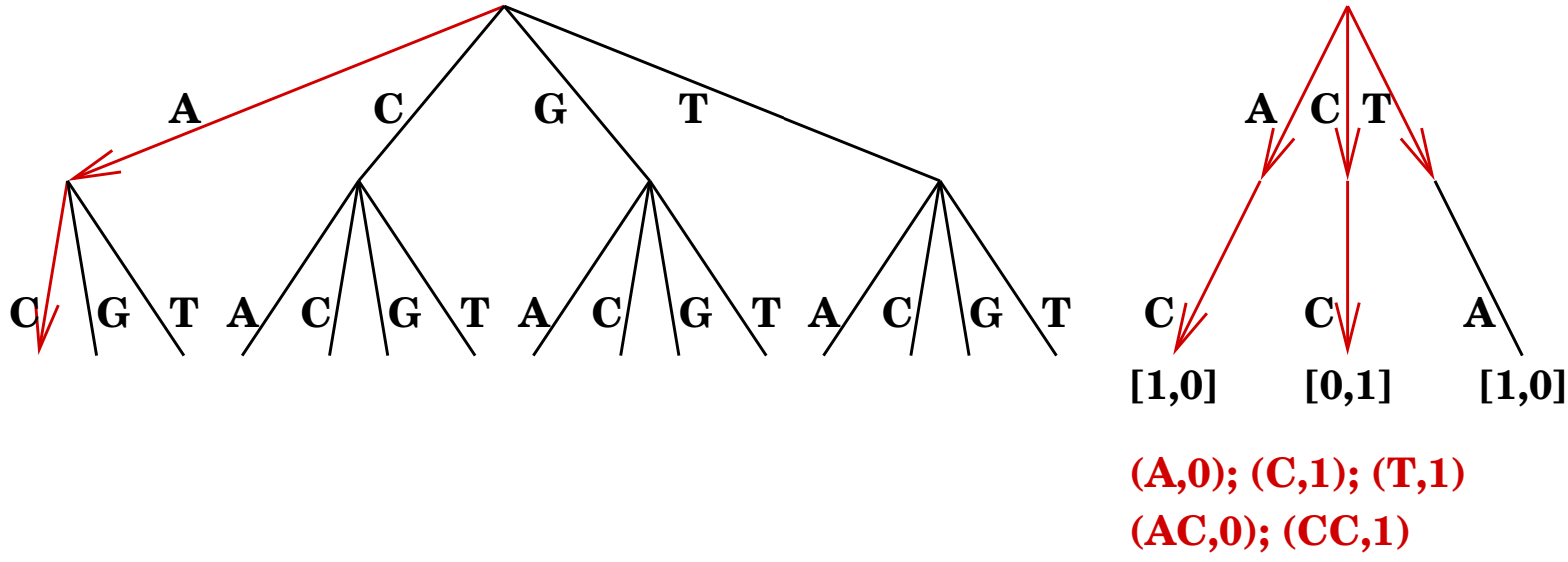


Extraction of Single Models

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

$k = 2$
$e = 1$
$q = 2$

Input sequences: TAC and CCC

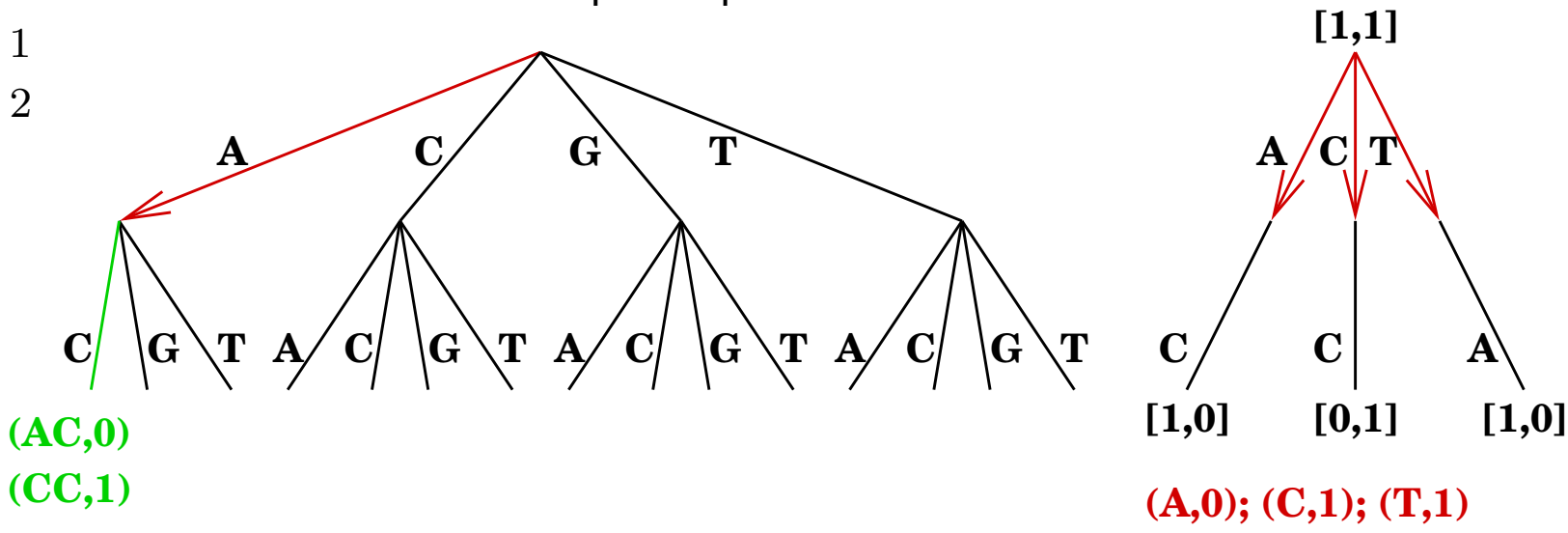


Extraction of Single Models

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

$k = 2$
$e = 1$
$q = 2$

Input sequences: TAC and CCC

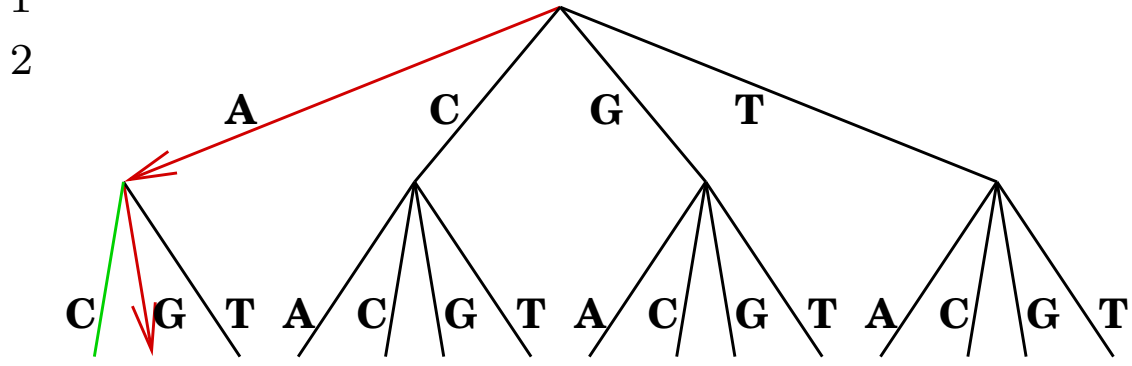


Extraction of Single Models

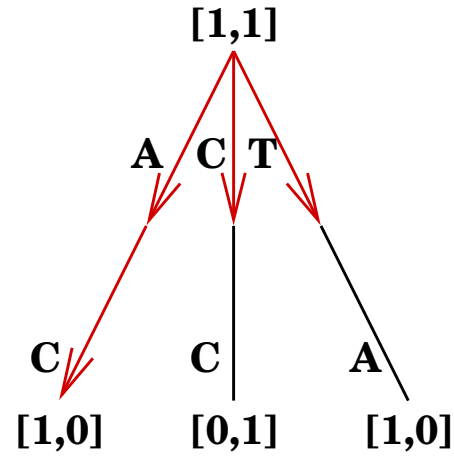
M.-F. Sagot, *Latin*, 1998

$k = 2$
$e = 1$
$q = 2$

Input sequences: TAC and CCC



(AC,0)
(CC,1)



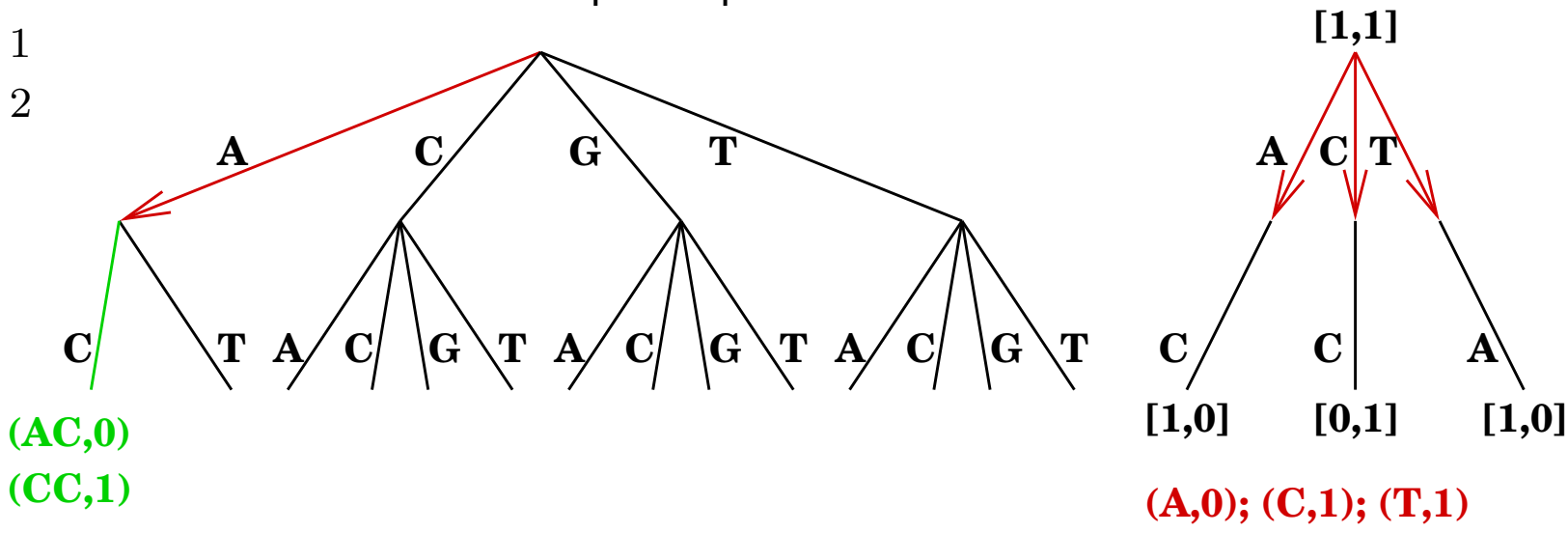
(A,0); (C,1); (T,1)
(AC,1)

Extraction of Single Models

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

$k = 2$
$e = 1$
$q = 2$

Input sequences: TAC and CCC

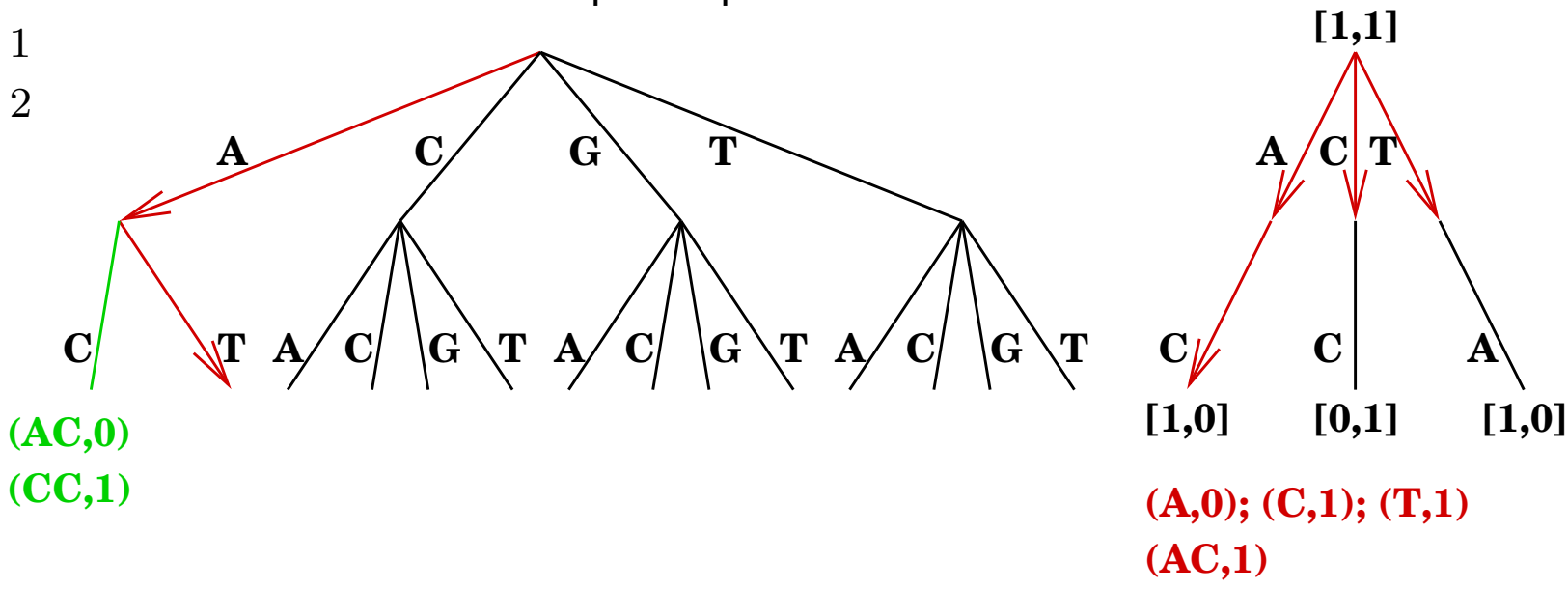


Extraction of Single Models

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

$k = 2$
$e = 1$
$q = 2$

Input sequences: TAC and CCC

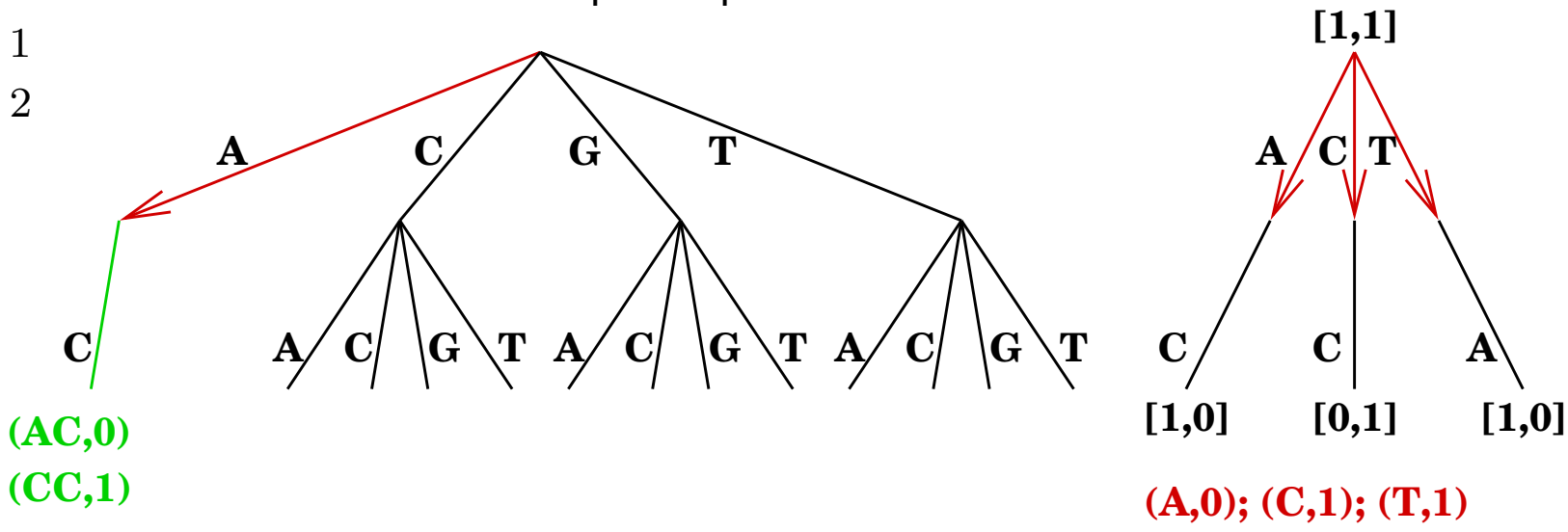


Extraction of Single Models

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

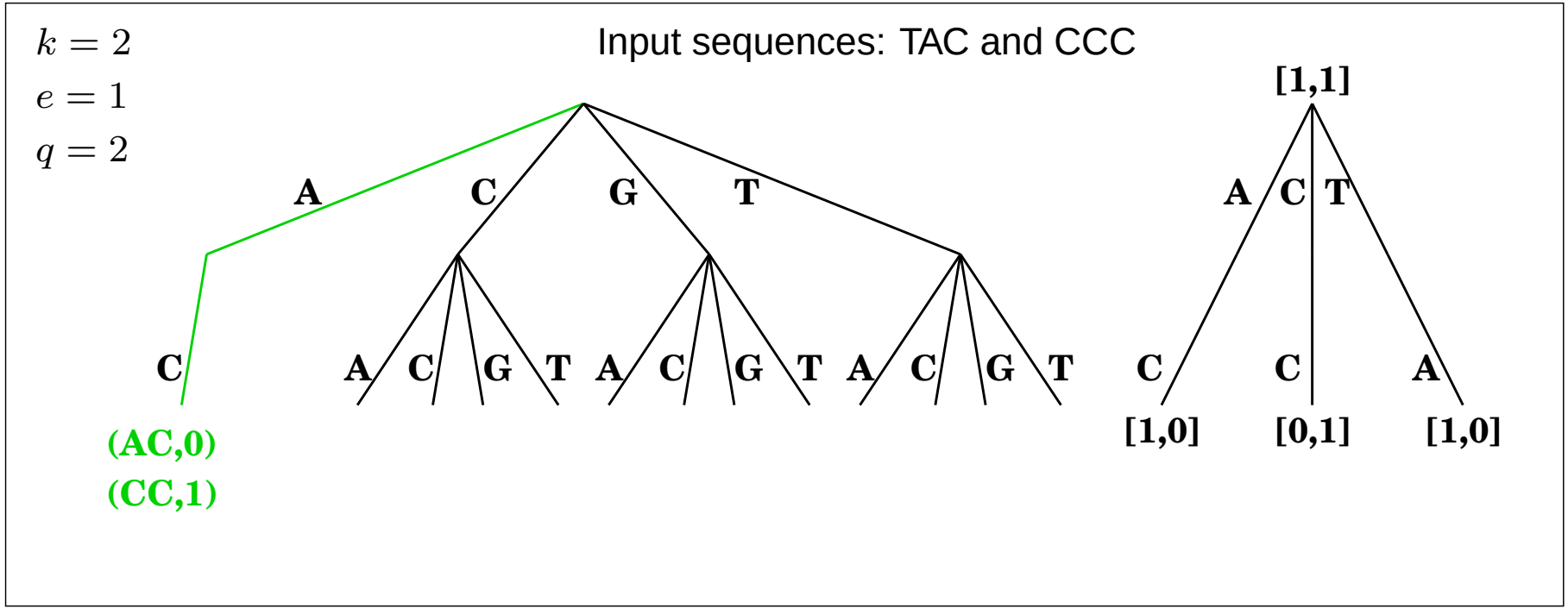
$k = 2$
$e = 1$
$q = 2$

Input sequences: TAC and CCC



Extraction of Single Models

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

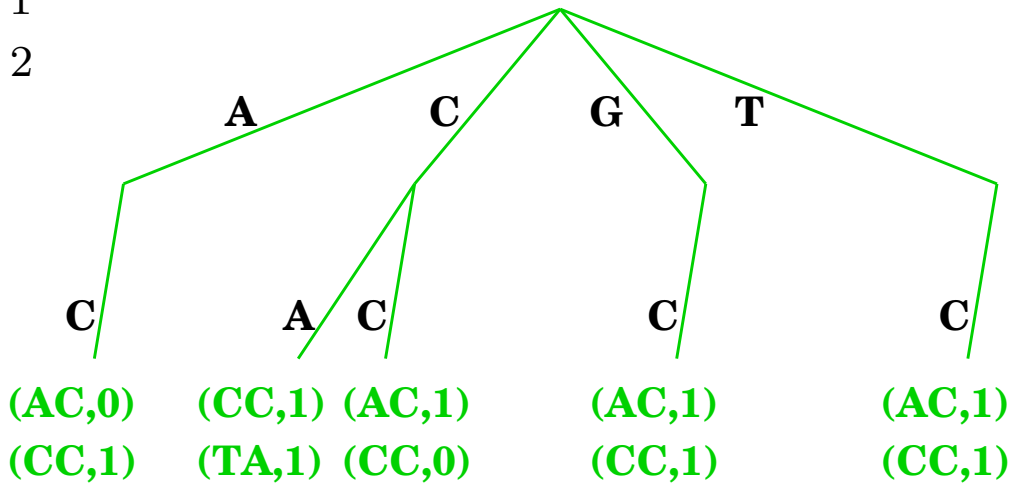


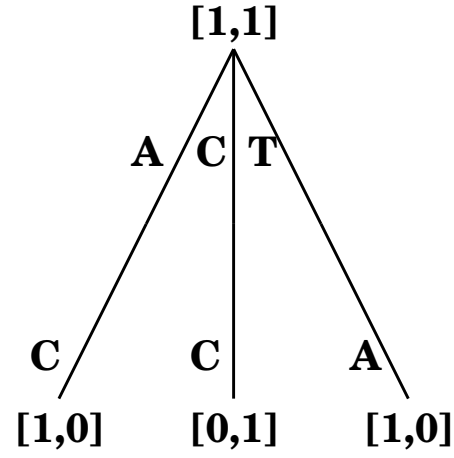
Extraction of Single Models

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

$k = 2$
$e = 1$
$q = 2$

Input sequences: TAC and CCC



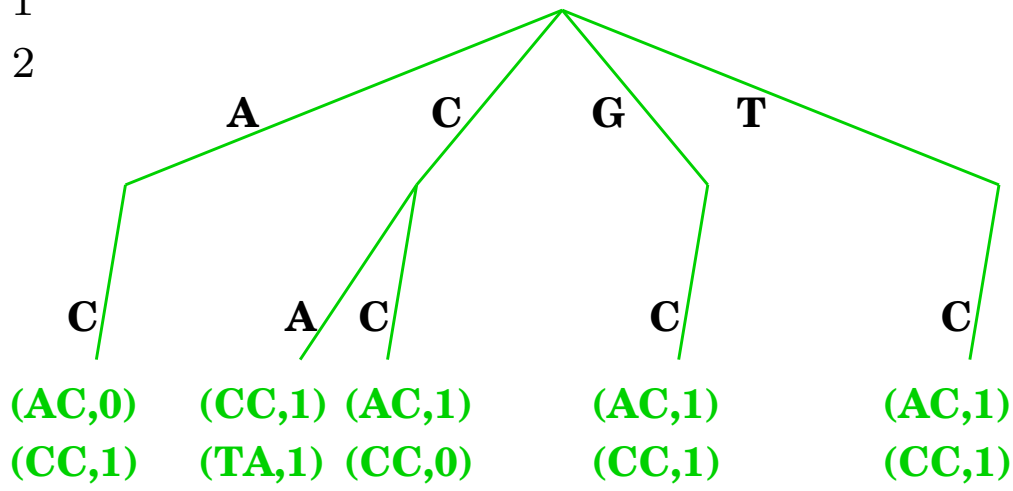


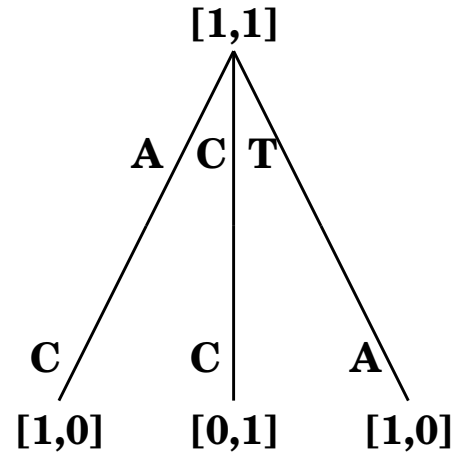
Extraction of Single Models

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

$k = 2$
$e = 1$
$q = 2$

Input sequences: TAC and CCC





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 $PotentialStarts$ 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 $PotentialStarts$
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 $PotentialStarts$
7. if (i < p)
8. **ExtractModels**(m = $m_1 \ldots m_i$,i + 1)
9. else
10. **KeepModel**($\langle (m_1, \ldots, m_p), ((d_{min_1}, d_{max_1}), \ldots, (d_{min_p}, d_{max_p})) \rangle$)

Extraction of Structured Models: SMILE

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

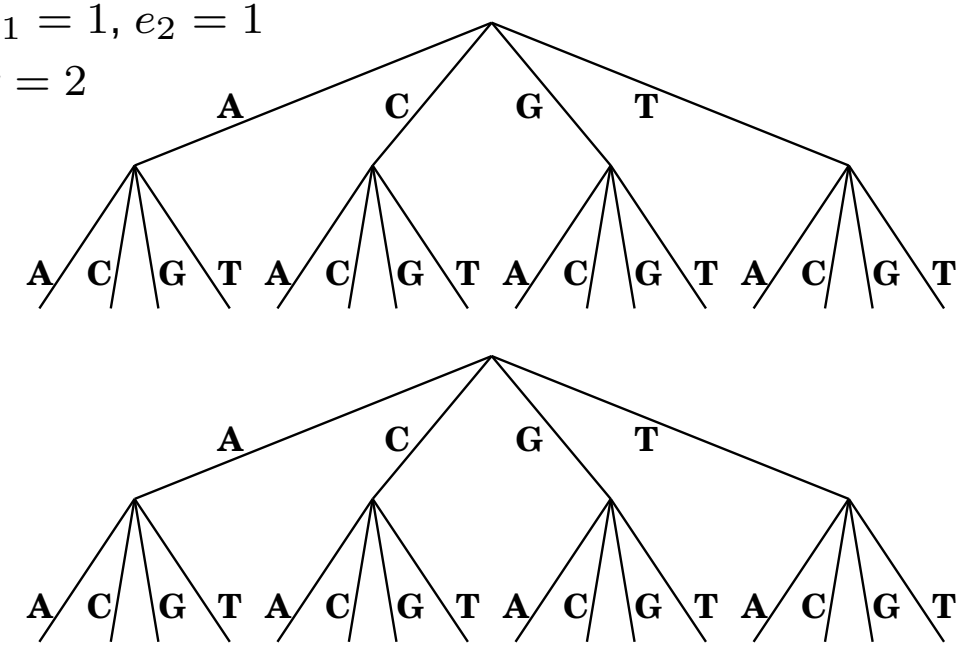
$p = 2$

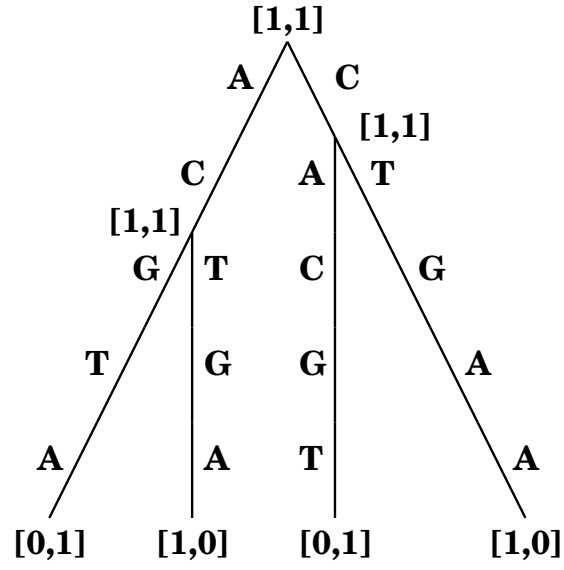
$k_1 = 2, d = 1, k_2 = 2$

$e_1 = 1, e_2 = 1$

$q = 2$

Input sequences: ACTGAA and CACGTA





Extraction of Structured Models: SMILE

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

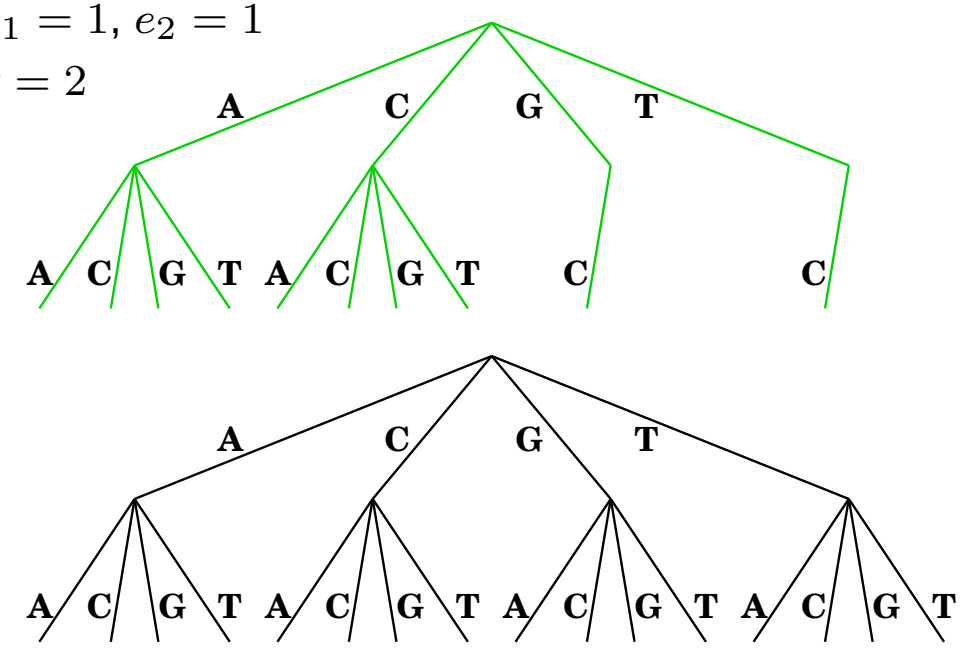
$p = 2$

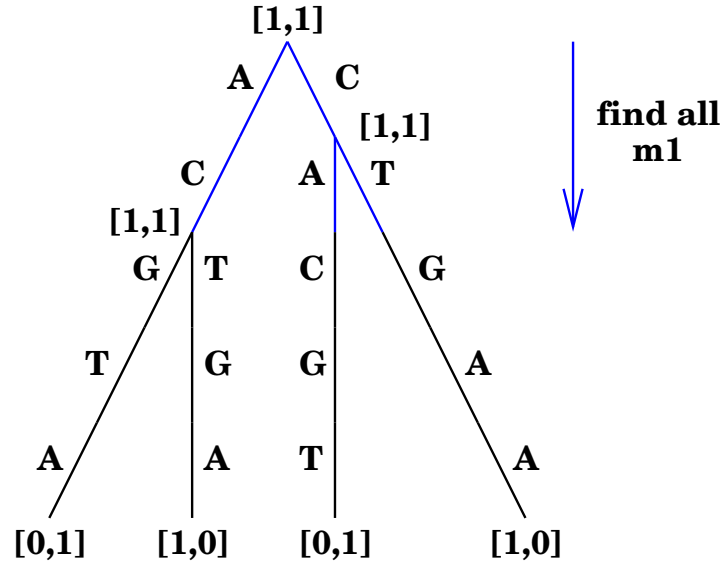
$k_1 = 2, d = 1, k_2 = 2$

$e_1 = 1, e_2 = 1$

$q = 2$

Input sequences: ACTGAA and CACGTA





Extraction of Structured Models: SMILE

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

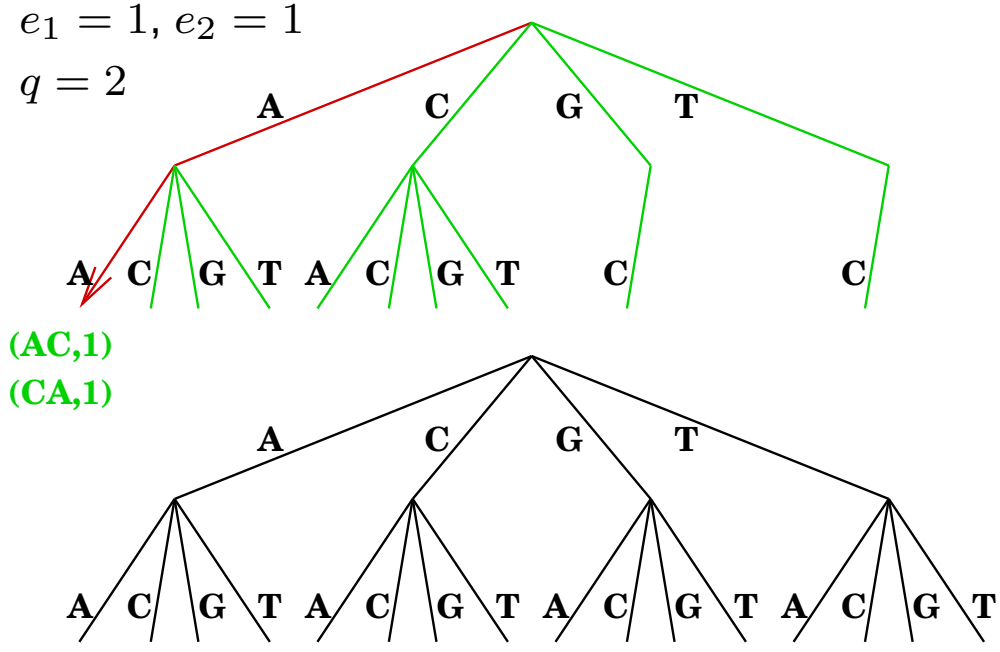
$p = 2$

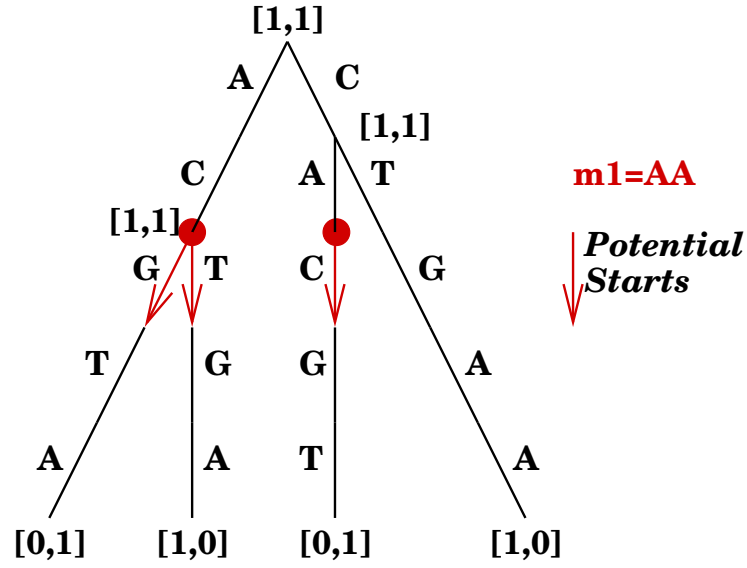
$k_1 = 2, d = 1, k_2 = 2$

$e_1 = 1, e_2 = 1$

$q = 2$

Input sequences: ACTGAA and CACGTA





Extraction of Structured Models: SMILE

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

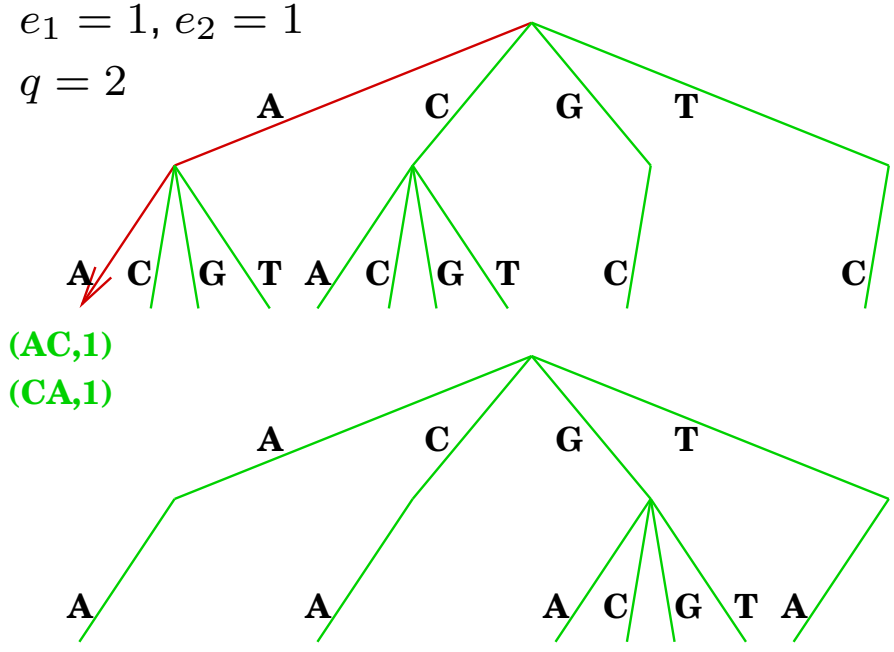
$p = 2$

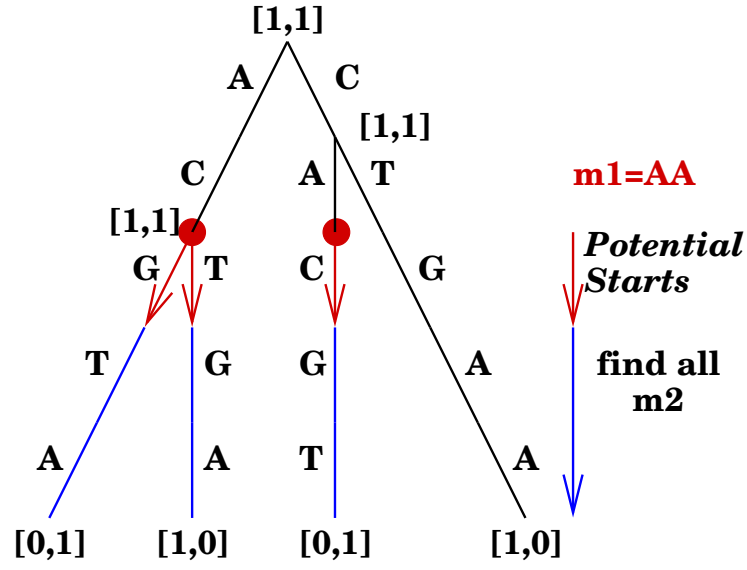
$k_1 = 2, d = 1, k_2 = 2$

$e_1 = 1, e_2 = 1$

$q = 2$

Input sequences: ACTGAA and CACGTA





Extraction of Structured Models: SMILE

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

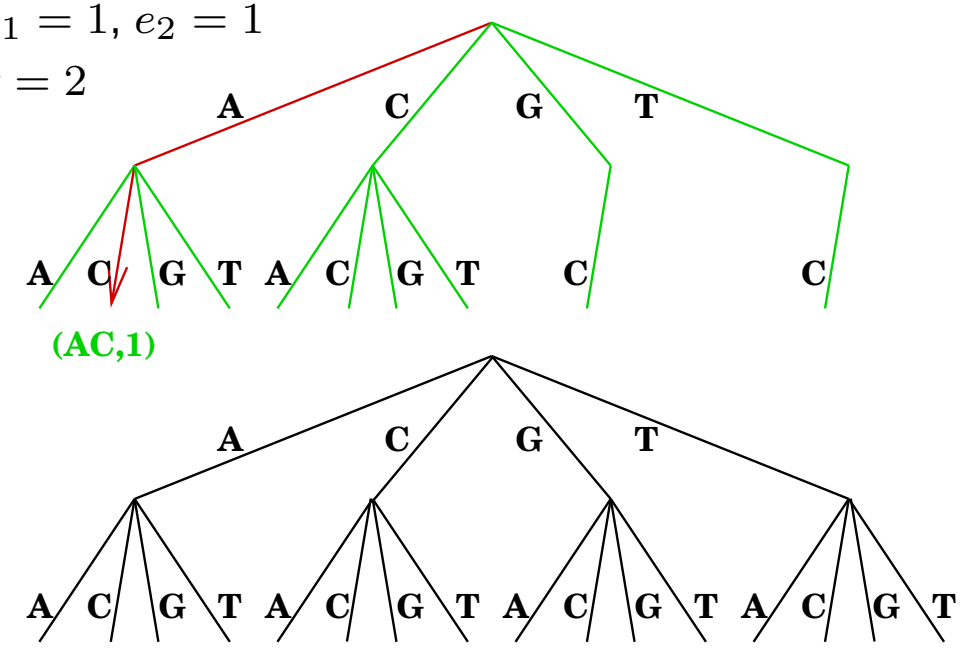
$p = 2$

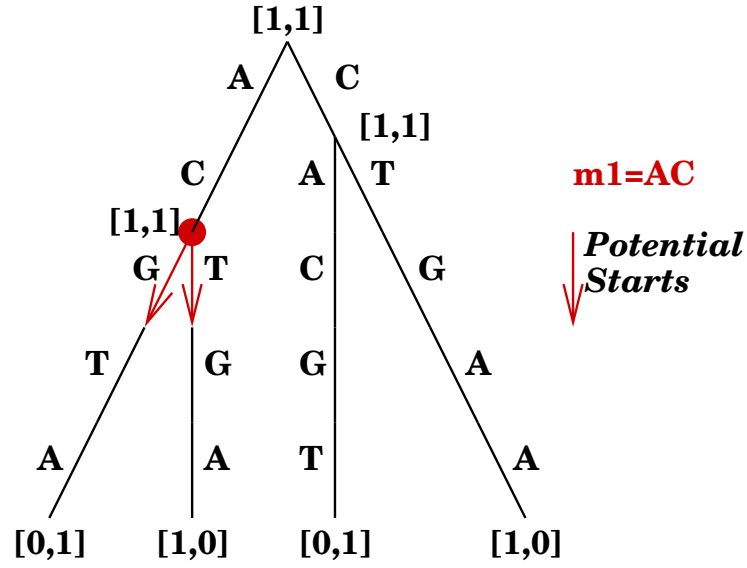
$k_1 = 2, d = 1, k_2 = 2$

$e_1 = 1, e_2 = 1$

$q = 2$

Input sequences: ACTGAA and CACGTA





Extraction of Structured Models: SMILE

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

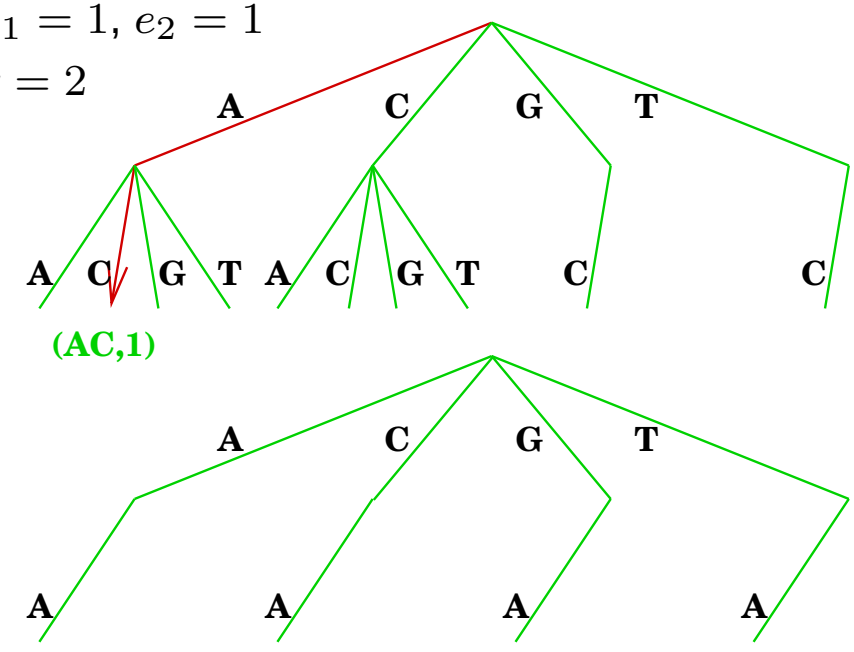
$p = 2$

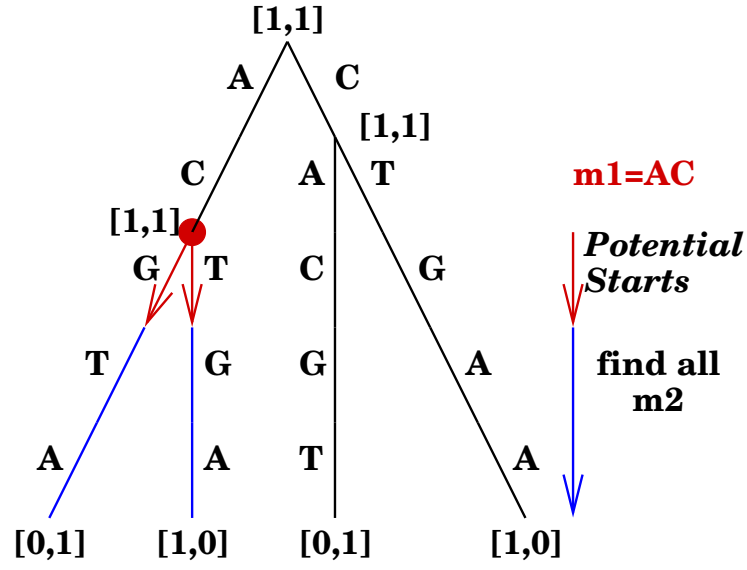
$k_1 = 2, d = 1, k_2 = 2$

$e_1 = 1, e_2 = 1$

$q = 2$

Input sequences: ACTGAA and CACGTA





Extraction of Structured Models: SMILE

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

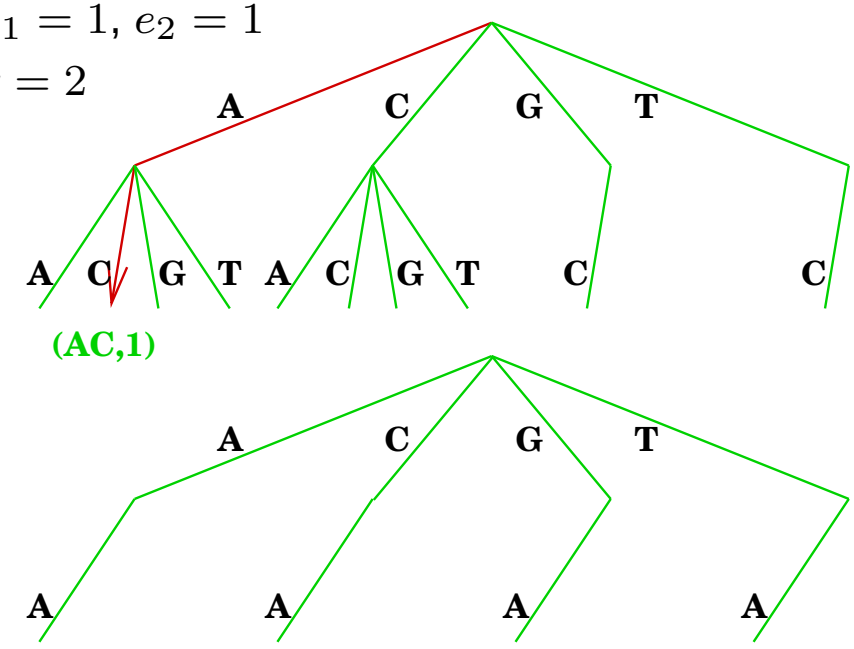
$p = 2$

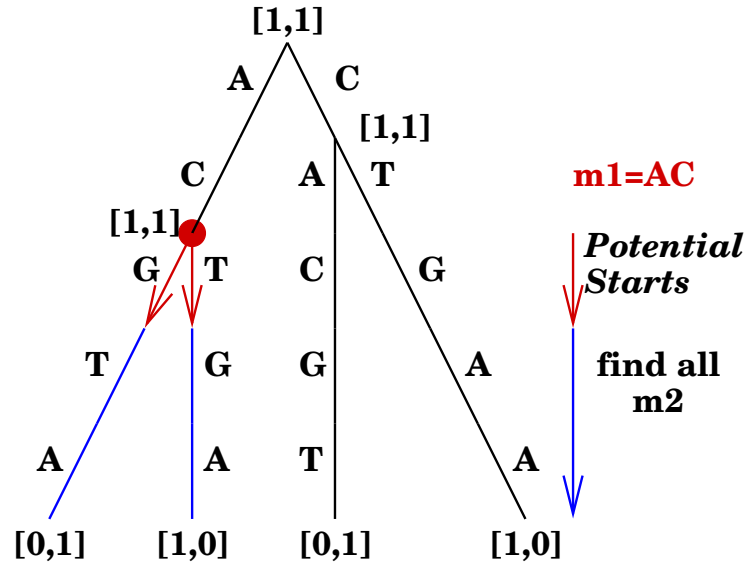
$k_1 = 2, d = 1, k_2 = 2$

$e_1 = 1, e_2 = 1$

$q = 2$

Input sequences: ACTGAA and CACGTA





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

PARTITION UP TO ε

PARTITION UP TO ε problem:

- ℓ gold bars
- $w_i \geq 0$ is the the weight of the ith 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 ε problem is NP-complete in the strong sense.

PARTITION UP TO ε

SimpleCut (**Partition** i, **GoldBars** ℓ, **Persons** r, **Weights** w_j, **CutFactor** c, **WorkOverload** ε)

1. find the smallest t such that $\frac{max\ \ w_j}{c^t} \leq \varepsilon$
2. for each $j \in \{1, ..., \ell\}$
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 $\gamma = w \times c^t \mod r$
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 ε

j	1	2
w_j	2	1

$r = 3 \;\; \varepsilon = 1$

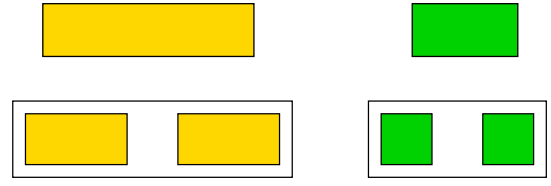
PARTITION UP TO ε

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 ε

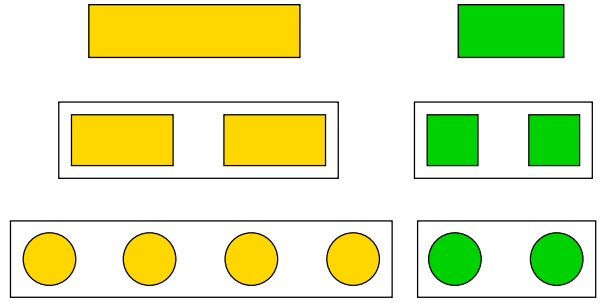
j	1	2
w_j	2	1

$r = 3 \;\; \varepsilon = 1$

$t = 1$

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

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



V1=[0,4) ***V2=[4,6)***

PARTITION UP TO ε

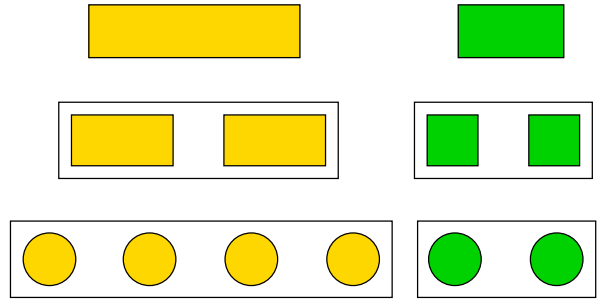
j	1	2
w_j	2	1

$r = 3 \;\; \varepsilon = 1$

$t = 1$

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

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



V1=[0,4) ***V2=[4,6)***

PARTITION UP TO ε

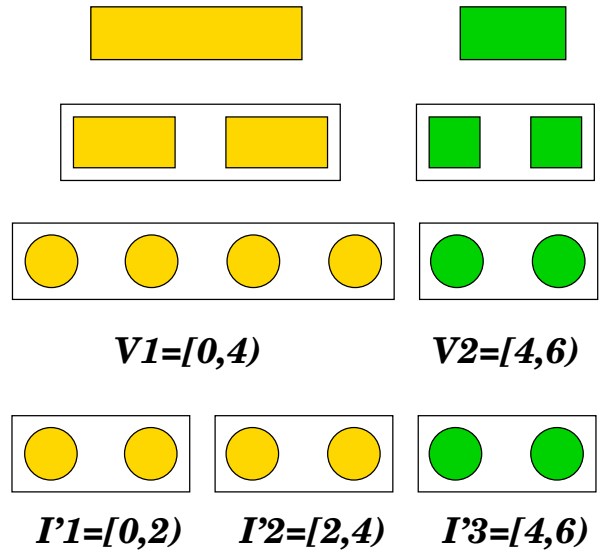
j	1	2
w_j	2	1

$r = 3 \;\; \varepsilon = 1$

$t = 1$

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

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



PARTITION UP TO ε

j	1	2
w_j	2	1

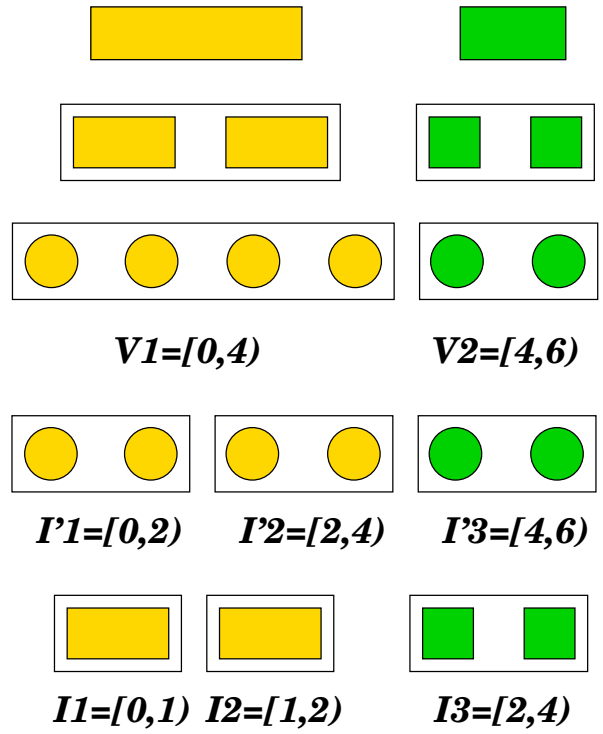
$r = 3 \;\; \varepsilon = 1$

$t = 1$

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

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

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



PARTITION UP TO ε

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 ε problem

Input of the SimpleCut algorithm for the ith 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
- $c = |\Sigma|$
- ε is an user parameter

Parallelization

Reducing the tree partition problem to the PARTITION UP TO ε problem

Input of the SimpleCut algorithm for the ith 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
- $c = |\Sigma|$
- ε is an user parameter

Output of the SimpleCut algorithm for the ith 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 ith grid node

Parallelization

Reducing the tree partition problem to the PARTITION UP TO ε problem

Input of the SimpleCut algorithm for the ith 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
- $c = |\Sigma|$
- ε is an user parameter

Output of the SimpleCut algorithm for the ith 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 ith grid node

SimpleCut (**Partition** i, **AlphabetSize** ℓ, **GridNodes** r, **Weights** w_j, **AlphabetSize** c, **WorkOverload** ε)

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

Parallelization

j	1	2	3	4
σ_j	A	C	T	G
w_j	2	1	1	2

$r = 5 \;\; \varepsilon = 1$

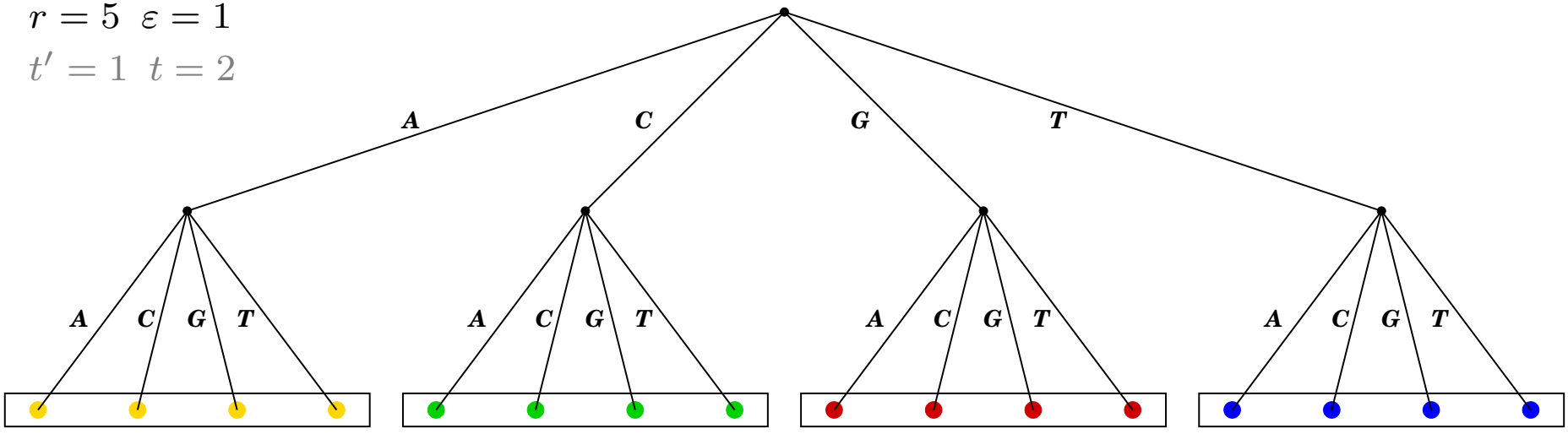
Parallelization

j	1	2	3	4
σ_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(depth($\mathcal{M}$) - 1 ,t')

$r = 5 \quad \varepsilon = 1$

$t' = 1 \quad t = 2$



Parallelization

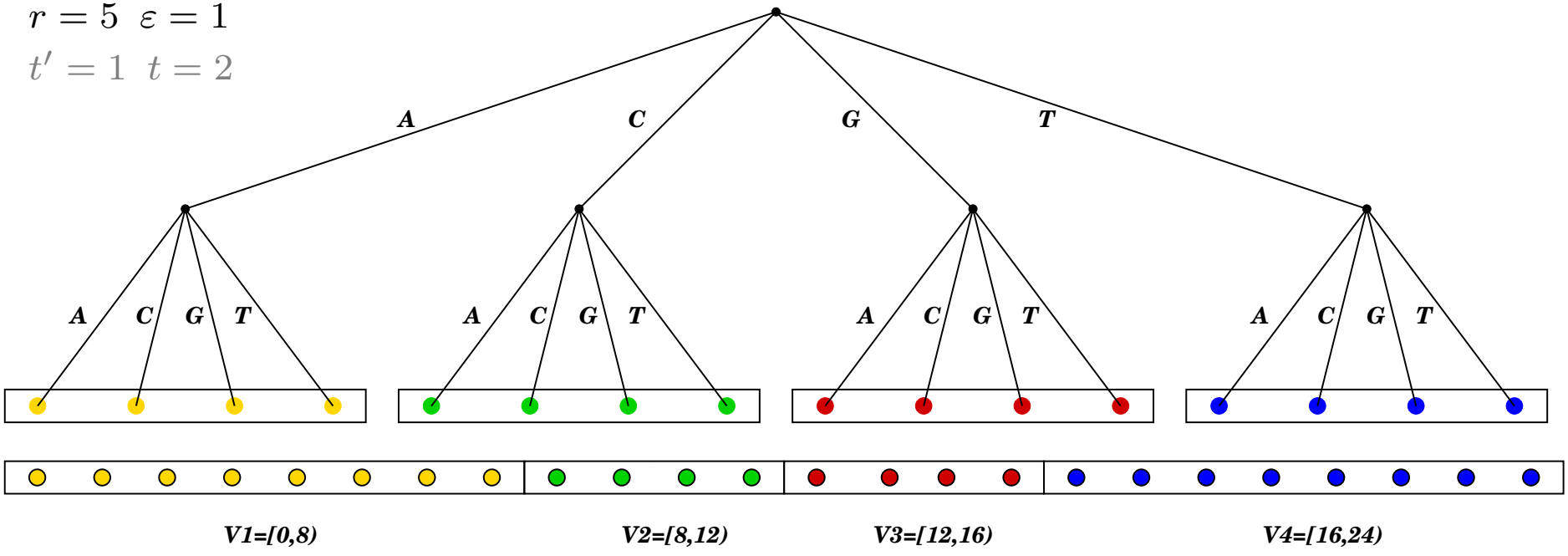
j	1	2	3	4
σ_j	A	C	T	G
w_j	2	1	1	2

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

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

$r = 5 \;\; \varepsilon = 1$

$t' = 1 \;\; t = 2$



Parallelization

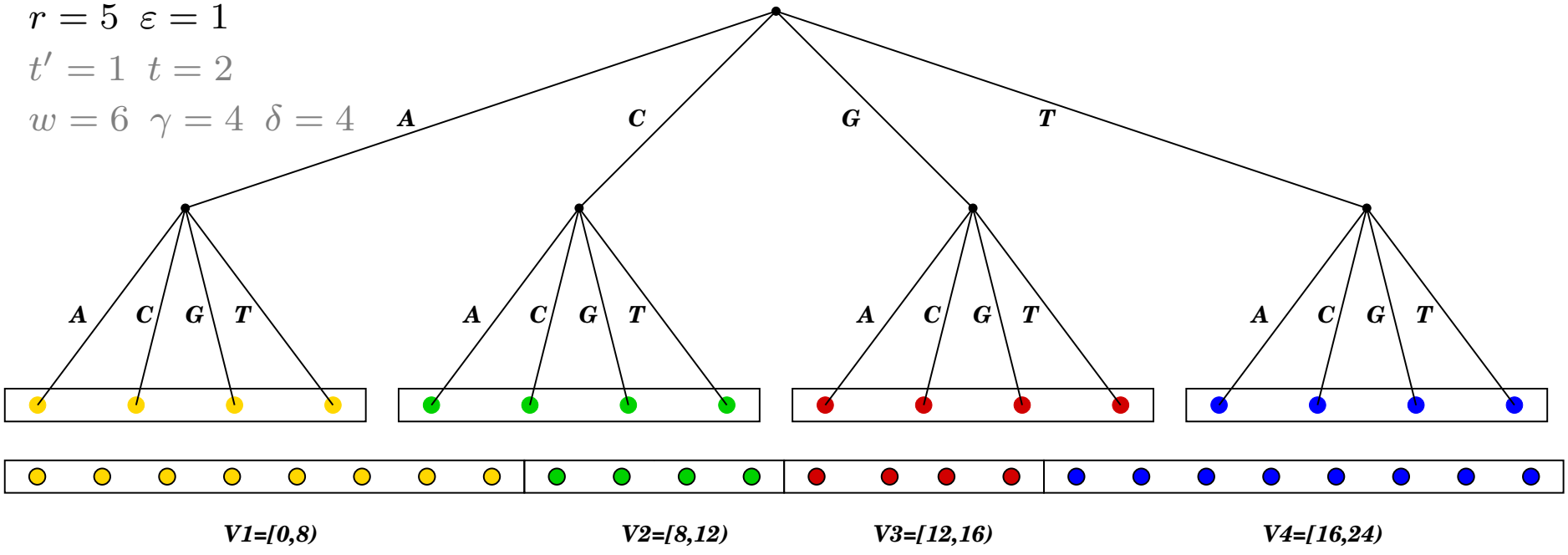
j	1	2	3	4
σ_j	A	C	T	G
w_j	2	1	1	2

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

$r = 5 \;\; \varepsilon = 1$

$t' = 1 \;\; t = 2$

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



Parallelization

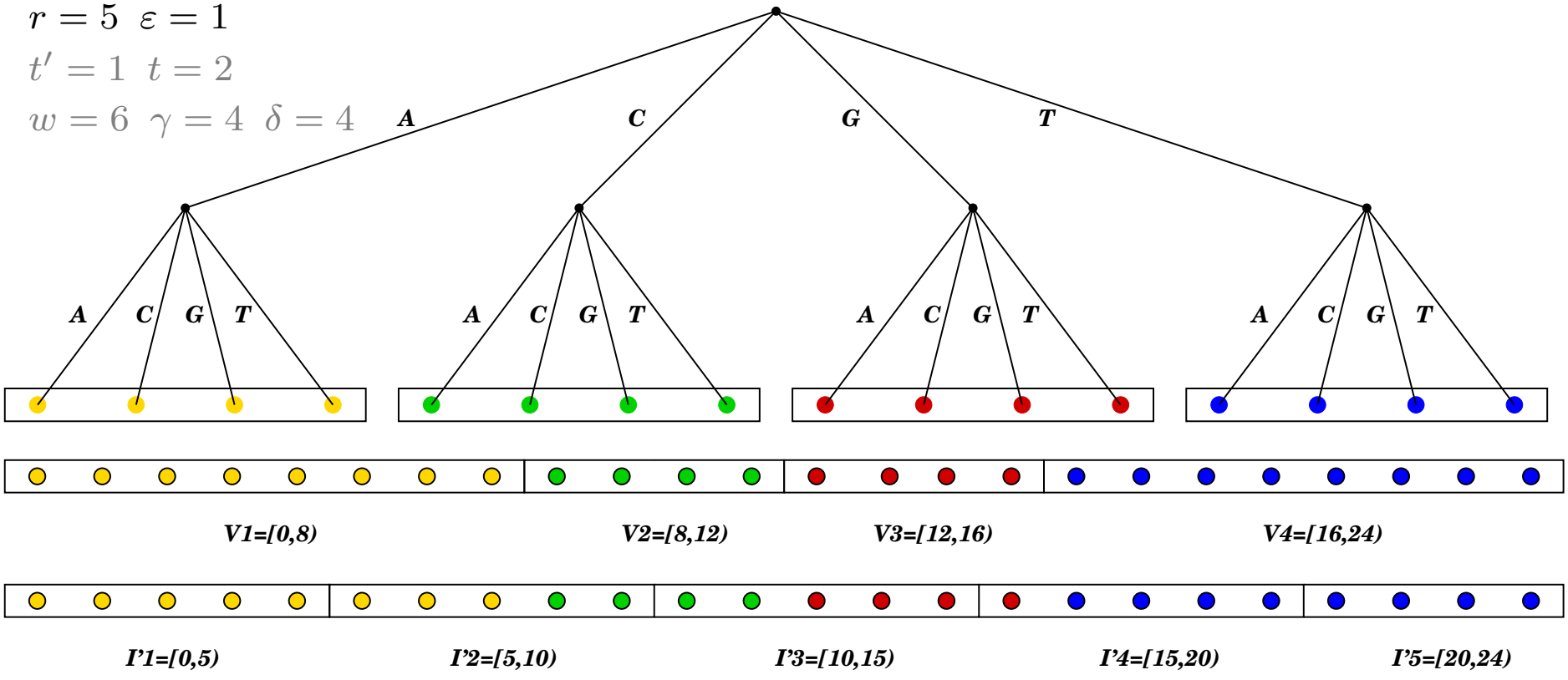
j	1	2	3	4
σ_j	A	C	T	G
w_j	2	1	1	2

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

$r = 5 \quad \varepsilon = 1$

$t' = 1 \quad t = 2$

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



Parallelization

j	1	2	3	4
σ_j	A	C	T	G
w_j	2	1	1	2

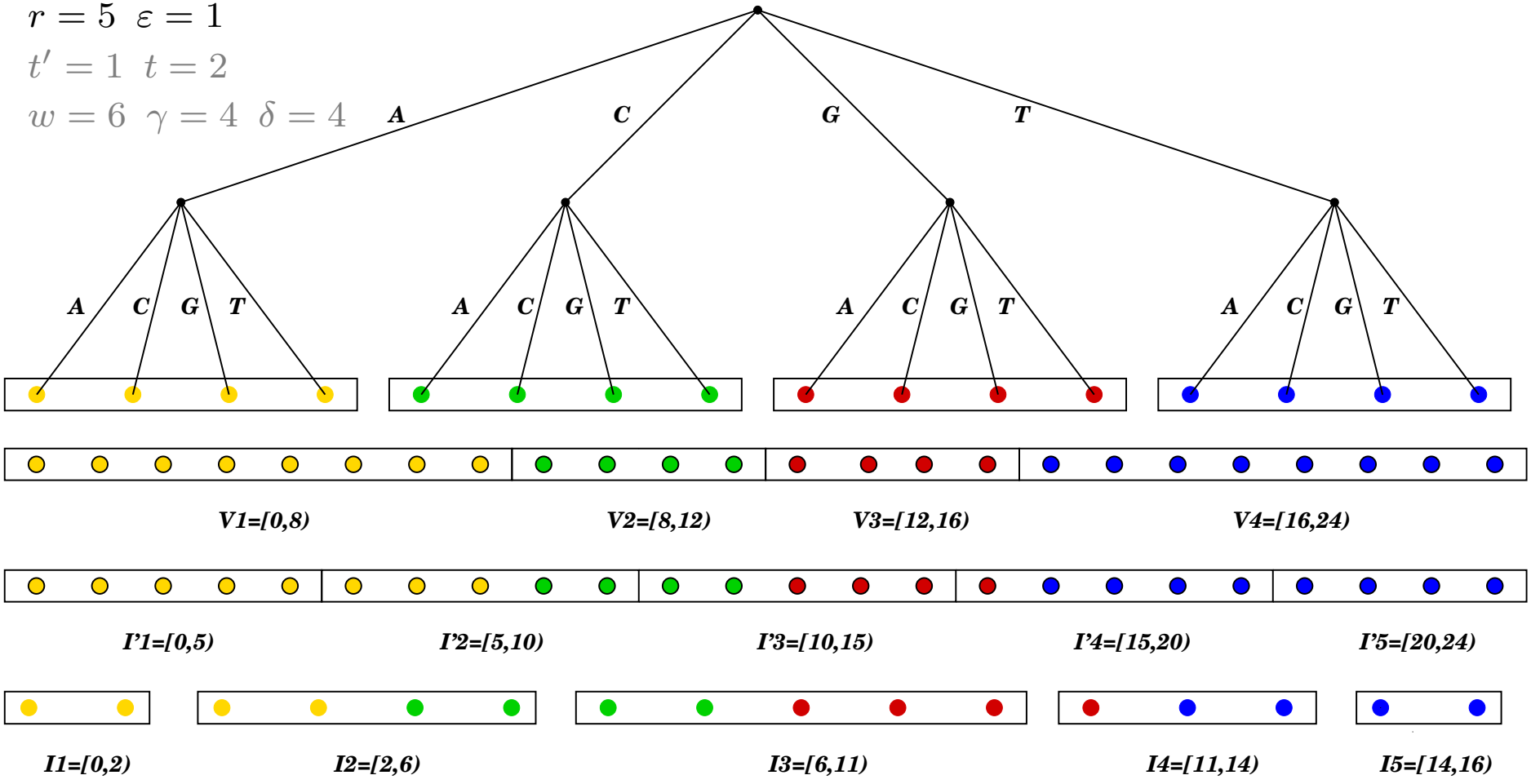
9. transform $I'_i = [a, b)$ into $I_i = [f(a), f(b))$ with

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

$r = 5 \;\; \varepsilon = 1$

$t' = 1 \;\; t = 2$

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



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 $PotentialStarts$ 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 $PotentialStarts$
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 $PotentialStarts$
7. if (i < p)
8. **PExtractModels**(m = $m_1 \ldots m_i$, i + 1, I_i)
9. else
10. **KeepModel**($\langle (m_1, \ldots, m_p), ((d_{min_1}, d_{max_1}), \ldots, (d_{min_p}, d_{max_p})) \rangle$)

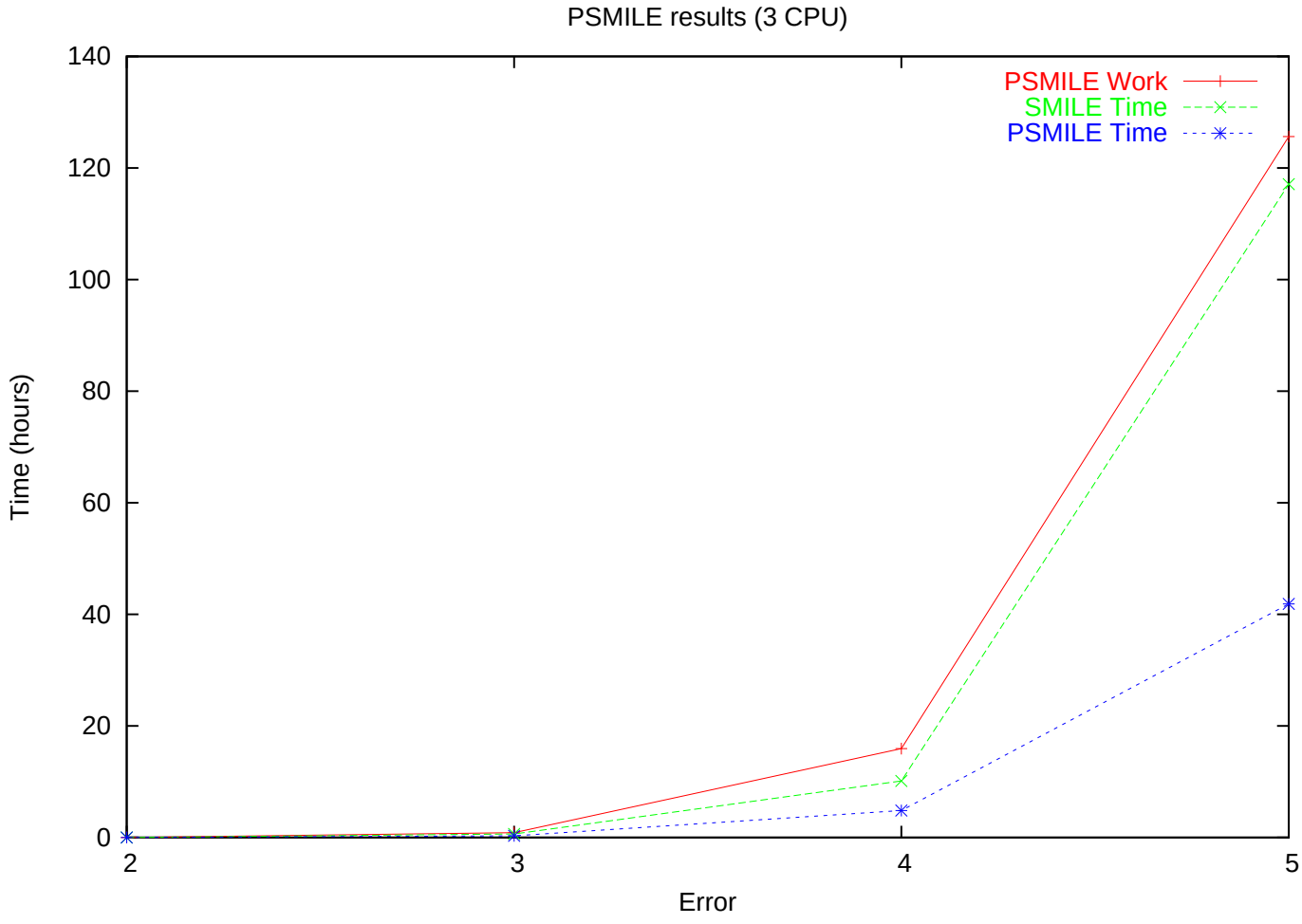
Parallelization

PSmile (**GridNode** i, **WorkOverload** ε)

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= SimpleCut$(i, |\Sigma|, r, (w_i)_{1 \leq i \leq |\Sigma|}, |\Sigma|, \varepsilon)$;
5. call PExtractModels($\mathcal{T}$, I_i);

Proposition. Assume Σ 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 $\frac{\varepsilon}{w} \leq \frac{1}{r}$.

Experimental results



Error	2	3	4	5
speed up	2.0	2.2	2.1	2.8

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

Appendix

Notation.

Π	Decision problems.
D_Π	Instance sets.
Y_Π	Yes sets.
$Length[I]$	Number of symbols used to describe the instance $I \in D_\Pi$.
$Max[I]$	Magnitude of the largest number in the instance $I \in D_\Pi$.

PARTITION problem:

- finite set A
- size $s(a) \in \mathbb{N}$ for each $a \in A$

Is there a subset $A' \subseteq A$ such that $\sum_{a' \in A'} s(a) = \sum_{a \in A \setminus A'} s(a)$?

Π	I	$Length[I]$	$Max[I]$
PRIMES	$x \in \mathbb{N}$	$\lceil \log_2(x) \rceil$	x
PARTITION	A finite set, $s(a) \in \mathbb{N}\ \forall_{a \in A}$	$\lvert A \rvert + \sum_{a \in A} \lceil \log_2(s(a)) \rceil$	$\max\{s(a) : a \in A\}$

back

Appendix

Definition. ***Polynomial***

An algorithm is polynomial if its time complexity is upper bounded by a polynomial function of the variable $Length[I]$.

Definition. ***NP***

Π is NP if there exists

- a set of witnesses W
- a polynomial algorithm $v : D_\Pi \times W \rightarrow \{0, 1\}$

such that:

- $I \in Y_\Pi \Rightarrow \exists_{w \in W}\ v(I, w) = 1$
- $I \notin Y_\Pi \Rightarrow \forall_{w \in W}\ v(I, w) = 0$

back

Appendix

Definition. ***Polynomial transformation***

A polynomial transformation from Π to Π' is a function $f : D_\Pi \to D_{\Pi'}$ such that:

- $\forall_{I \in D_\Pi} I \in Y_\Pi \Leftrightarrow f(I) \in Y_{\Pi'}$
- f can be computed in polynomial time

Definition. ***NP hard***

Π is NP hard if for any NP problem Π' there exists a polynomial transformation from Π' to Π.

Definition. ***NP complete***

A decision problem Π is NP complete if

- Π is NP
- Π is NP-hard

back

Appendix

Definition. ***Number problem***
Π is a number problem if $\nexists_p \ \forall_{I \in D_\Pi} Max[I] \leq p(Length[I])$.

Definition. Π_p
$\Pi_p = \{I \in \Pi : Max[I] \leq p(Length[I])\}$.

Definition. ***NP complete in the strong sense***
Π is NP complete in the strong sense if

- Π is NP
- $\exists_p \ \Pi_p$ is NP complete

Definition. ***Pseudo-polynomial***
An algorithm is pseudo-polynomial if its time complexity is upper bounded by a polynomial function of the two variables $Length[I]$ and $Max[I]$.

Proposition. A NP complete problem in the strong sense cannot be solved by a pseudo-polynomial time algorithm, unless P=NP.

back

Appendix

Definition. ***Pseudo-polynomial transformation***

A pseudo-polynomial transformation from Π to Π' is a function $f : D_\Pi \rightarrow D_{\Pi'}$ such that:

- $\forall_{I \in D_\Pi} I \in Y_\Pi \Leftrightarrow f(I) \in Y_{\Pi'}$
- f can be computed in pseudo-polynomial time
- $\exists_{p_1} \forall_{I \in D_\Pi} p_1(Length'[f(I)]) \geq Length[I]$
- $\exists_{p_2} \forall_{I \in D_\Pi} Max'[f(I)] \leq p_2(Max[I], Length[I])$

Proposition. If

- Π is NP complete in the strong sense
- Π' is NP
- exists a pseudo-polynomial transformation from Π to Π'

then Π' is NP complete in the strong sense.

back

Appendix

3-PARTITION problem:

- finite set A with $3m$ elements
- bound $B \in \mathbb{N}$
- size $s(a) \in \mathbb{N}$ for each $a \in A$ such that:
  - $B/4 < s(a) < B/2$
  - $\sum_{a \in A} s(a) = mB$

Can A be partitioned into m disjoint sets $S_1, S_2, \ldots, S_m$ such that for all $1 \leq i \leq m$

$$\sum_{a \in S_i} s(a) = B?$$

Proposition. The **3-PARTITION** problem is NP complete in the strong sense.

back

Appendix

Definition. ***Ratio bound***

An approximation algorithm, for a minimization problem, has a ratio bound of $\rho(x)$ if

$$\forall_{\text{input } x} \ \frac{C(x)}{C^*(x)} \leq \rho(x)$$

where:

- $C(x)$ is the cost of the solution produced by the algorithm
- $C^*(x)$ is the cost of the optimal solution

back

Appendix

Definition. ***Work***

The work performed by a parallel algorithm P, for an input x, is defined as

$$W_P(x) = \text{time of } P(x) \times \text{number of processing units.}$$

Definition. ***Work-efficient***

A parallel algorithm P is work-efficient with respect to the sequential version S when

$$\exists_C \ \forall_{\text{input } x} \ W_p(x) \leq C \times \text{ time of } S(x).$$

back