Minimal absent words in a sliding window & applications to on-line pattern matching

Maxime Crochemore[1,2], **Alice Héliou**[3], Gregory Kucherov[2], Laurent Mouchard[4], Solon Pissis[1], Yann Ramusat[5]

[1] Department of Informatics, King's College London, London, UK

[2] CNRS & Université Paris-Est

[3] LIX, Ecole Polytechnique, CNRS, INRIA, Université Paris-Saclay

[4] University of Rouen, LITIS EA 4108, TIBS, Rouen

[5] DI ENS, CNRS, PSL Research University & INRIA Paris

11 septembre 2017 – FCT Bordeaux

Definition : Minimal Absent Word

A minimal absent word of a sequence is an absent word whose proper factors (longest prefix, and longest suffix) all occur in the sequence.

An upper bound on the number of minimal absent words is $\mathcal{O}(\sigma n)$.

Crochemore et al. 1998, Mignosi et al. 2002

$$\begin{array}{c} \overset{0}{} \overset{1}{} \overset{2}{} \overset{3}{} \overset{4}{} \overset{5}{} \overset{6}{} \overset{7}{} \\ S = \mathrm{A\,C\,A\,C\,A\,A\,G\,C} \end{array}$$

Definition : Minimal Absent Word

A minimal absent word of a sequence is an absent word whose proper factors (longest prefix, and longest suffix) all occur in the sequence.

An upper bound on the number of minimal absent words is $\mathcal{O}(\sigma n)$.

Crochemore et al. 1998, Mignosi et al. 2002

$$\overset{0}{A}\ \overset{1}{C}\ \overset{2}{A}\ \overset{3}{C}\ \overset{4}{A}\ \overset{5}{A}\ \overset{6}{G}\ \overset{7}{C}$$

$$S = ACACAAGC$$

AAA, AAC, CACAC, CAG, CC, CG, GA, GCA, GG

Definition : Minimal Absent Word

A minimal absent word of a sequence is an absent word whose proper factors (longest prefix, and longest suffix) all occur in the sequence.

An upper bound on the number of minimal absent words is $\mathcal{O}(\sigma n)$.

Crochemore et al. 1998, Mignosi et al. 2002

$$\overset{0}{A}\ \overset{1}{C}\ \overset{2}{A}\ \overset{3}{C}\ \overset{4}{A}\ \overset{5}{A}\ \overset{6}{G}\ \overset{7}{C}$$

$$S = ACACAAGC$$

AAA, AAC, CACAC, CAG, CC, CG, GA, GCA, GG

Definition : Minimal Absent Word

A minimal absent word of a sequence is an absent word whose proper factors (longest prefix, and longest suffix) all occur in the sequence.

An upper bound on the number of minimal absent words is $\mathcal{O}(\sigma n)$.

Crochemore et al. 1998, Mignosi et al. 2002

$$\overset{0}{S}\overset{1}{=}\overset{2}{A}\overset{3}{C}\overset{4}{A}\overset{5}{C}\overset{6}{A}\overset{7}{A}\overset{8}{G}\overset{9}{C}$$

AAA, AAC, CACAC, CAG, CC, CG, GA, GCA, GG

Definition : Minimal Absent Word

A minimal absent word of a sequence is an absent word whose proper factors (longest prefix, and longest suffix) all occur in the sequence.

An upper bound on the number of minimal absent words is $\mathcal{O}(\sigma n)$.

Crochemore et al. 1998, Mignosi et al. 2002

$$\overset{0}{A}\ \overset{1}{C}\ \overset{2}{A}\ \overset{3}{C}\ \overset{4}{A}\ \overset{5}{A}\ \overset{6}{G}\ \overset{7}{C}$$

$$S = ACACAAGC$$

AAA, AAC, CACAC, CAG, CC, CG, GA, GCA, GG

Definition : Minimal Absent Word

A minimal absent word of a sequence is an absent word whose proper factors (longest prefix, and longest suffix) all occur in the sequence.

An upper bound on the number of minimal absent words is $\mathcal{O}(\sigma n)$.

Crochemore et al. 1998, Mignosi et al. 2002

$$S = \overset{0}{A}\,\overset{1}{C}\,\overset{2}{A}\,\overset{3}{C}\,\overset{4}{A}\,\overset{5}{A}\,\overset{6}{G}\,\overset{7}{C}$$

AAA, AAC, CACAC, CAG, CC, CG, GA, GCA, GG

Definition : Minimal Absent Word

A minimal absent word of a sequence is an absent word whose proper factors (longest prefix, and longest suffix) all occur in the sequence.

An upper bound on the number of minimal absent words is $\mathcal{O}(\sigma n)$.

Crochemore et al. 1998, Mignosi et al. 2002

$$\overset{0}{A}\,\overset{1}{C}\,\overset{2}{A}\,\overset{3}{C}\,\overset{4}{A}\,\overset{5}{A}\,\overset{6}{G}\,\overset{7}{C}$$

S=ACACACAAGC

AAA, AAC, CACACC, CAG, CC, CG, GA, GCA, GG

Definition : Minimal Absent Word

A minimal absent word of a sequence is an absent word whose proper factors (longest prefix, and longest suffix) all occur in the sequence.

An upper bound on the number of minimal absent words is $\mathcal{O}(\sigma n)$.

Crochemore et al. 1998, Mignosi et al. 2002

$$S = \overset{0}{A}\,\overset{1}{C}\,\overset{2}{A}\,\overset{3}{C}\,\overset{4}{A}\,\overset{5}{A}\,\overset{6}{G}\,\overset{7}{C}$$

AAA, AAC, CACAC, CAG, CC, CG, GA, GCA, GG

Applications

Biology

- 3 sequences (TTTCGCCCGACT, TACGCCCTATCG, CCTACGCGCAAA), found in Ebola genomes as coding for proteins are absent from the Human genome.

Applications

Biology

- 3 sequences (TTTCGCCCGACT, TACGCCCTATCG, CCTACGCGCAAA), found in Ebola genomes as coding for proteins are absent from the Human genome.

BioInformatics

- Metric based on minimal absent words
  $\rightarrow$ Phylogeny (Chairungsee et al., 2012, Crochemore et al, 2016).

Applications

Biology

- 3 sequences (TTTCGCCCGACT, TACGCCCTATCG, CCTACGCGCAAA), found in Ebola genomes as coding for proteins are absent from the Human genome.

BioInformatics

- Metric based on minimal absent words
  $\rightarrow$ Phylogeny (Chairungsee et al., 2012, Crochemore et al, 2016).

Computer Science

- Data compression using anti-dictionnaries (Crochemore et al., 2000, Fiala and Holub, 2008).

Definition : Maximal repeated pair

A maximal repeated pair in a S is a triple (i, j, w) such that :

- w occurs in S at positions i and j
- $S[i-1] \neq S[j-1]$
- $S[i+|w|] \neq S[j+|w|]$

Definition : Maximal repeated pair

A maximal repeated pair in a S is a triple (i, j, w) such that :

- w occurs in S at positions i and j
- $S[i-1] \neq S[j-1]$
- $S[i+|w|] \neq S[j+|w|]$

Lemma

If awb is a minimal absent word of S, then there exist positions i and j such that (i, j, w) is a maximal repeated pair of S.

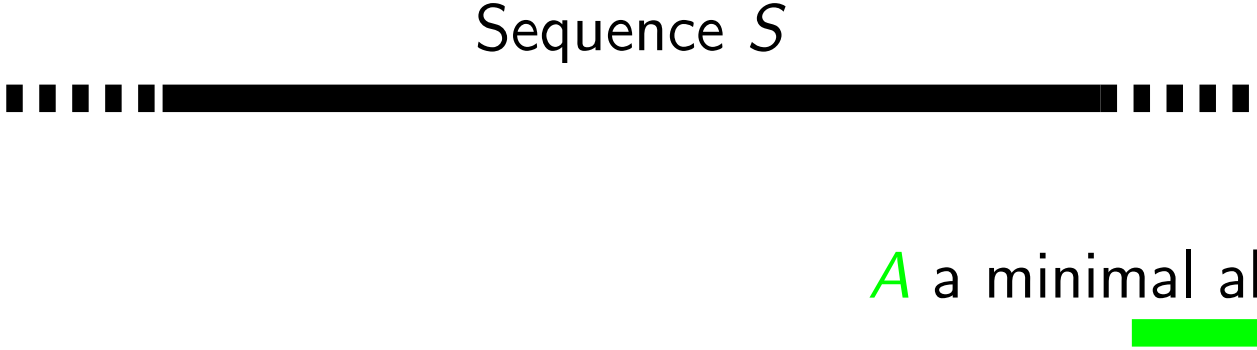
Definition : Maximal repeated pair

A maximal repeated pair in a S is a triple (i, j, w) such that :

- w occurs in S at positions i and j
- $S[i-1] \neq S[j-1]$
- $S[i+|w|] \neq S[j+|w|]$

Lemma

If awb is a minimal absent word of S, then there exist positions i and j such that (i, j, w) is a maximal repeated pair of S.



Definition : Maximal repeated pair

A maximal repeated pair in a S is a triple (i, j, w) such that :

- w occurs in S at positions i and j
- $S[i-1] \neq S[j-1]$
- $S[i+|w|] \neq S[j+|w|]$

Lemma

If awb is a minimal absent word of S, then there exist positions i and j such that (i, j, w) is a maximal repeated pair of S.

A a minimal absent word of S

longest prefix of A

Definition : Maximal repeated pair

A maximal repeated pair in a S is a triple (i, j, w) such that :

- w occurs in S at positions i and j
- $S[i-1] \neq S[j-1]$
- $S[i+|w|] \neq S[j+|w|]$

Lemma

If awb is a minimal absent word of S, then there exist positions i and j such that (i, j, w) is a maximal repeated pair of S.

A a minimal absent word of S

longest suffix of A

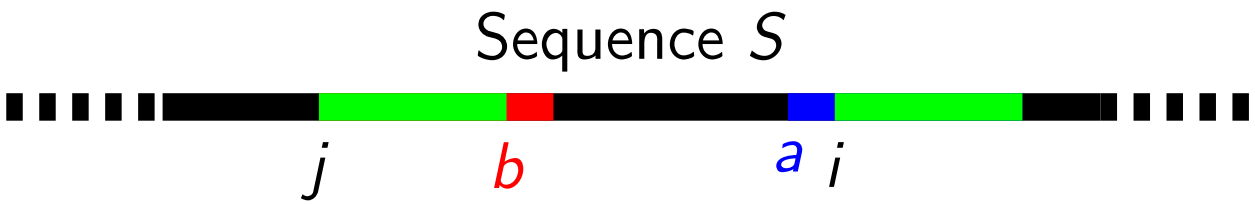
Definition : Maximal repeated pair

A maximal repeated pair in a S is a triple (i, j, w) such that :

- w occurs in S at positions i and j
- $S[i-1] \neq S[j-1]$
- $S[i+|w|] \neq S[j+|w|]$

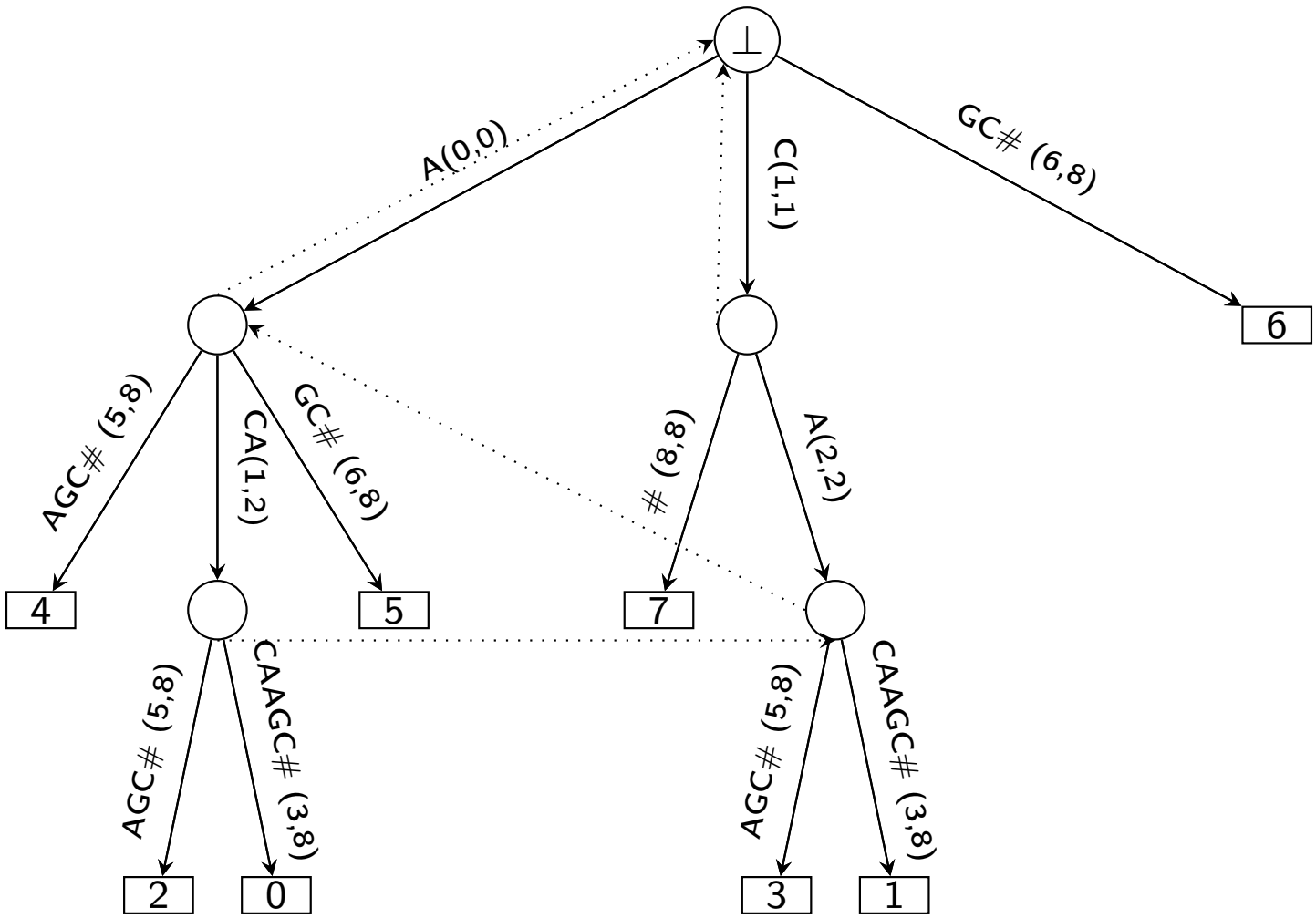
Lemma

If awb is a minimal absent word of S, then there exist positions i and j such that (i, j, w) is a maximal repeated pair of S.

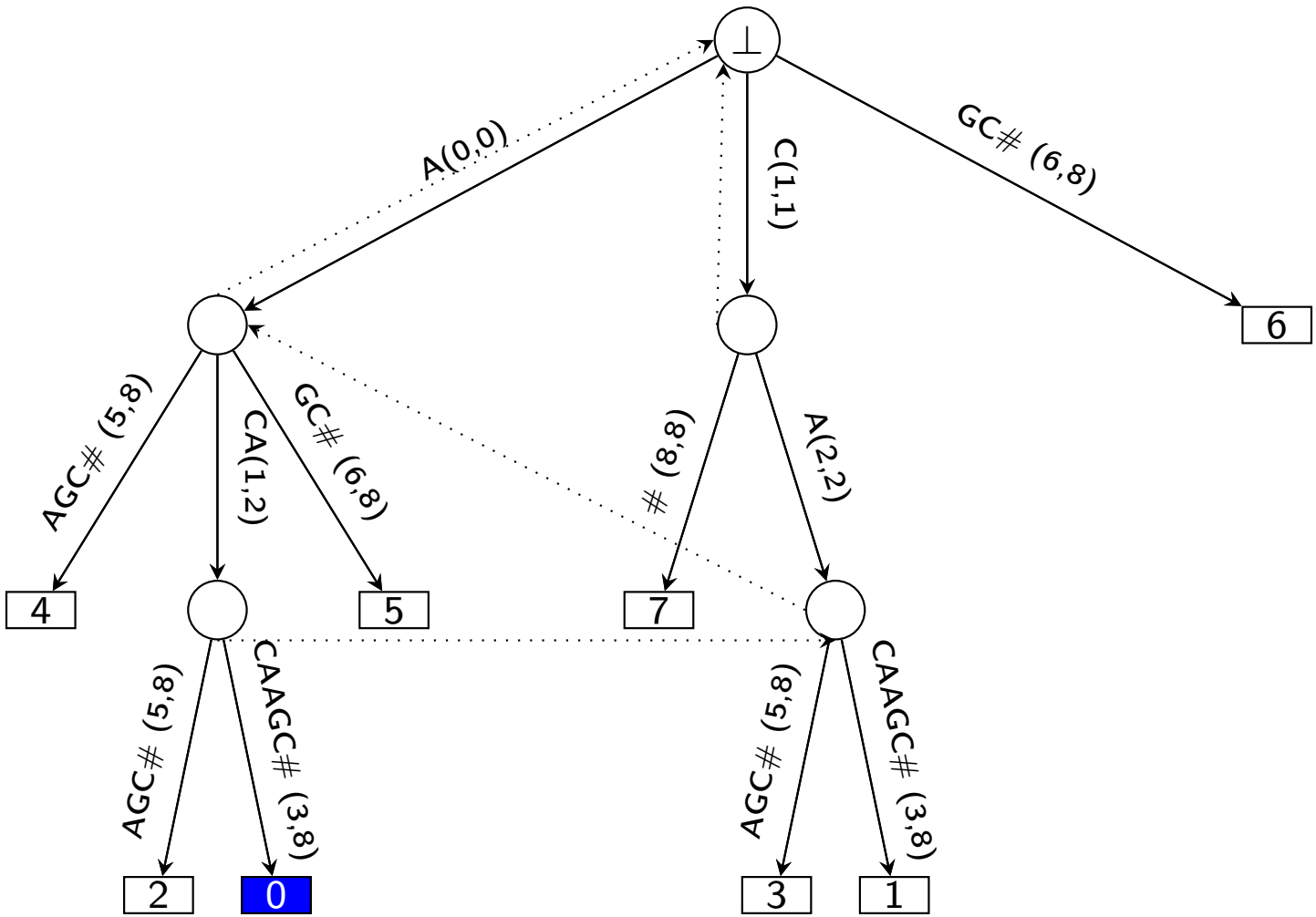


A a minimal absent word of S

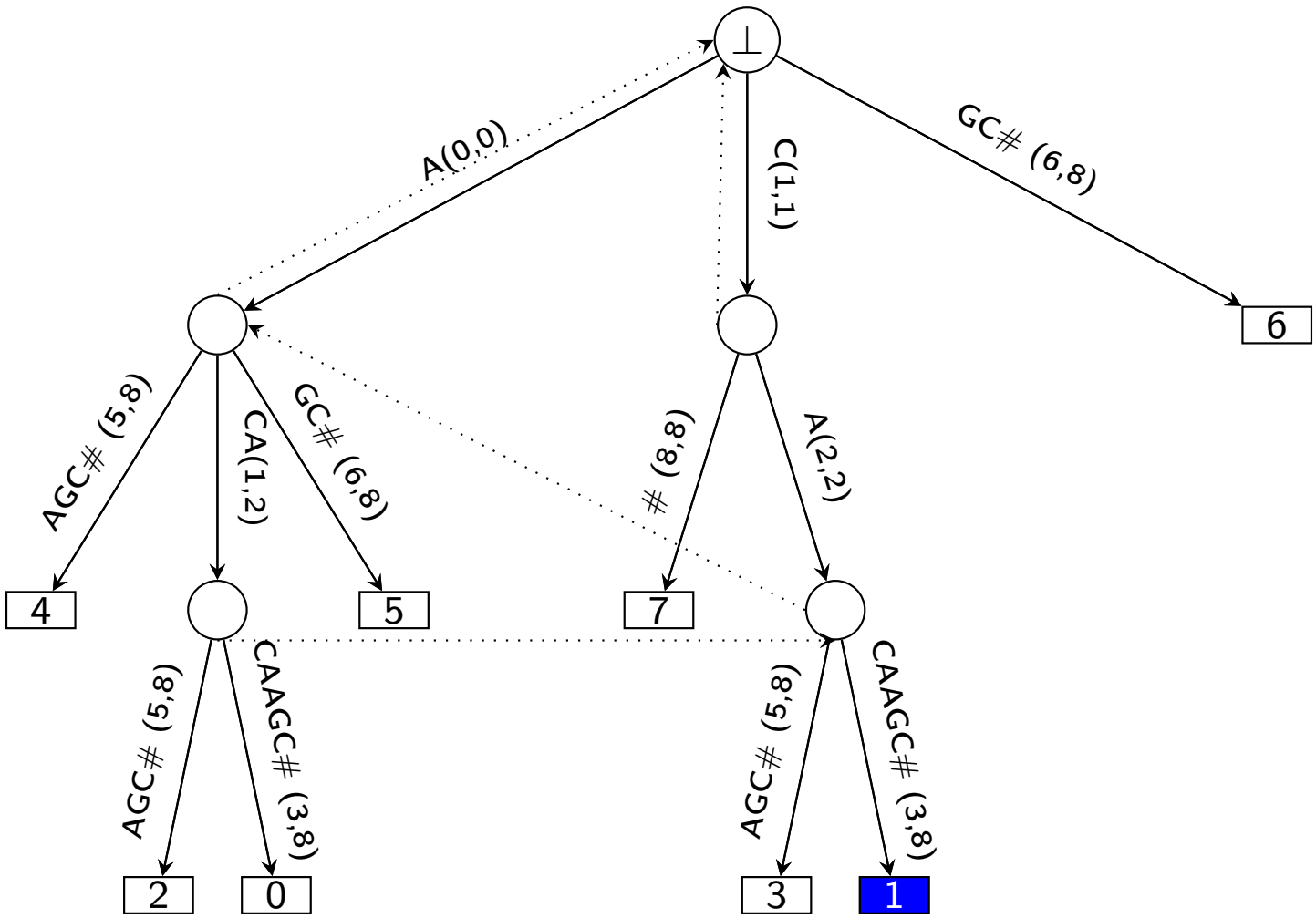
Suffix tree of $S =$ 0 1 2 3 4 5 6 7 8
A C A C A A G C #



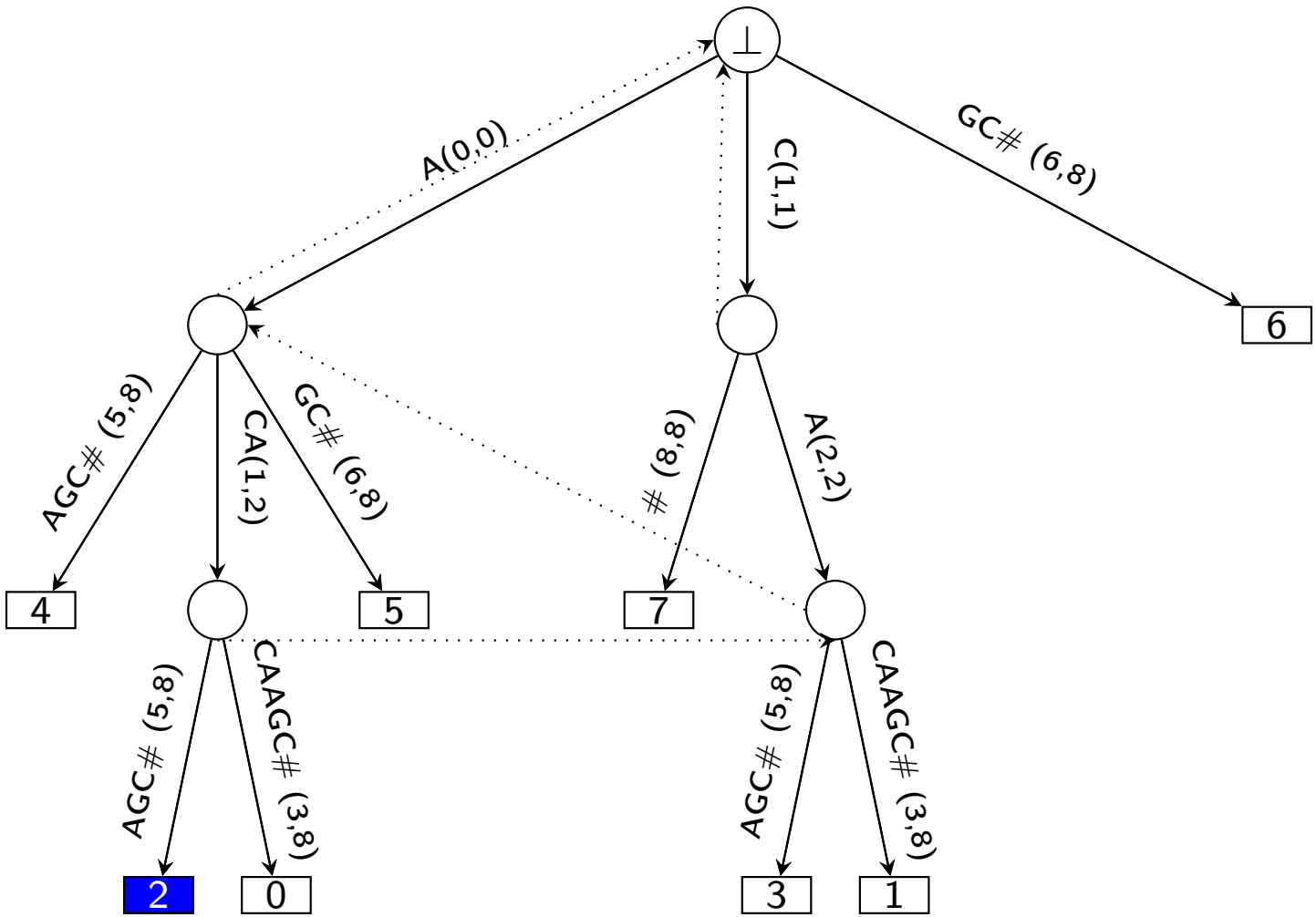
Suffix tree of S = A C A C A A G C # (0 1 2 3 4 5 6 7 8)



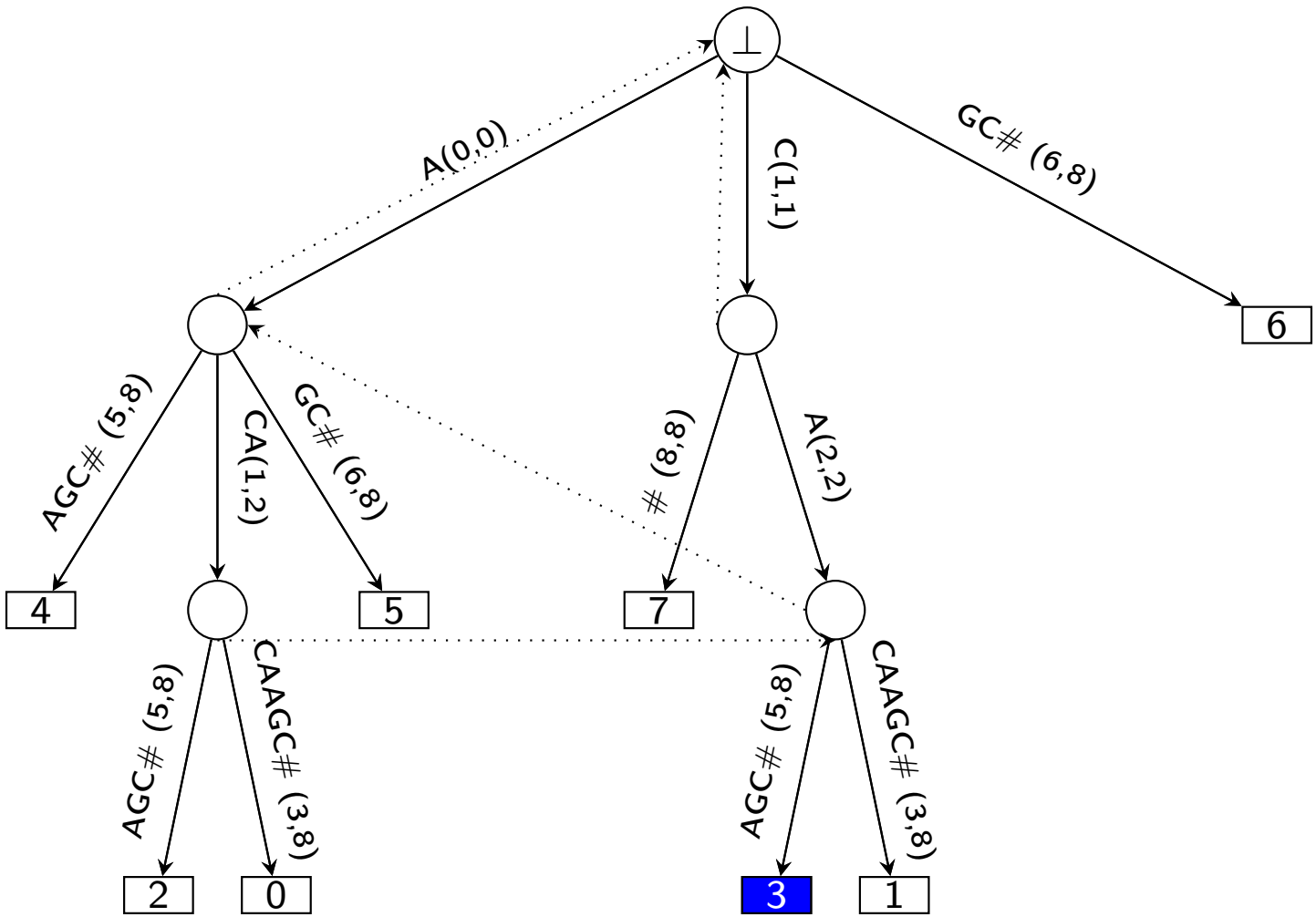
Suffix tree of S = 0 1 2 3 4 5 6 7 8
A C A C A A G C #



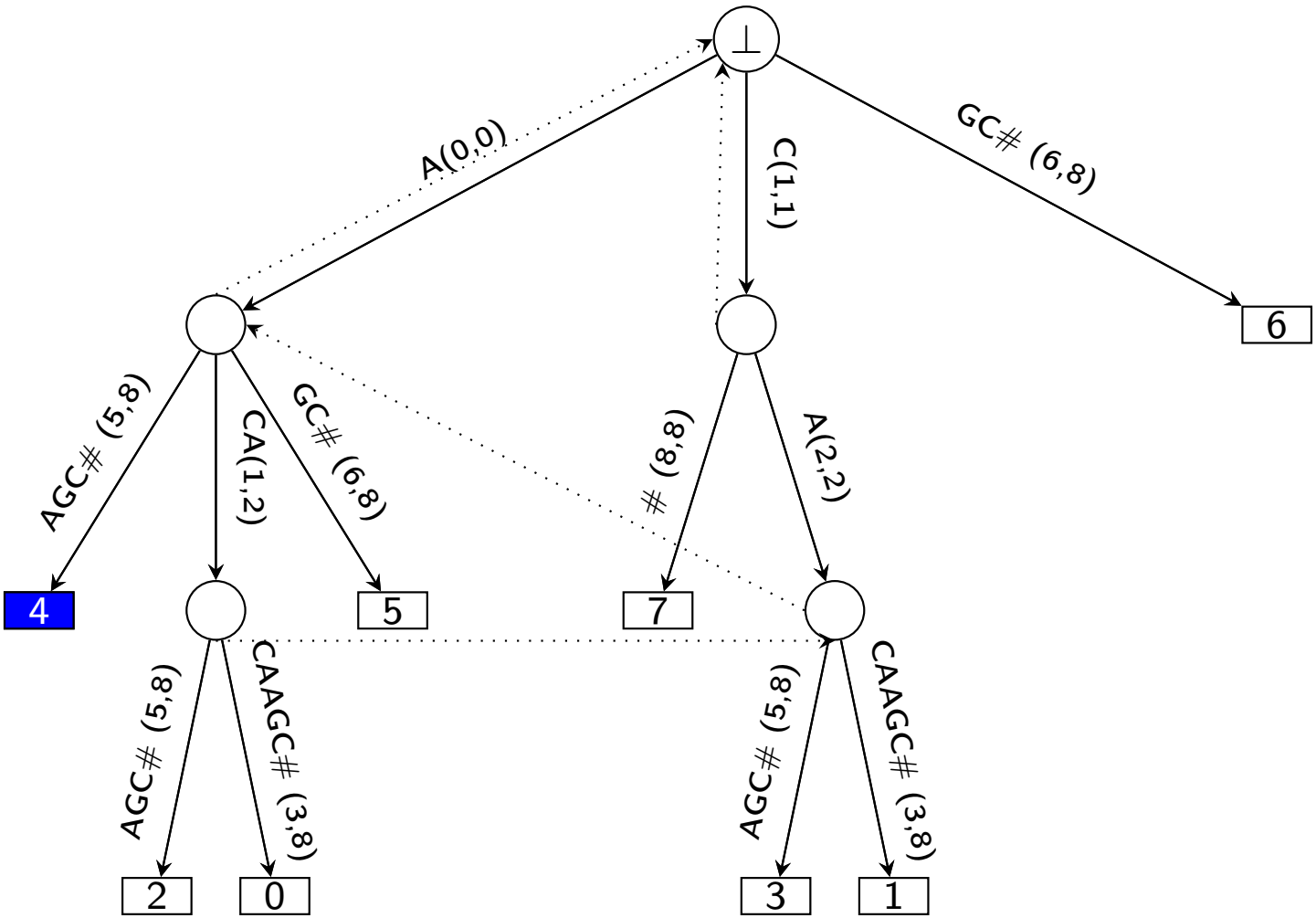
Suffix tree of S = A C A C A A G C # (0 1 2 3 4 5 6 7 8)



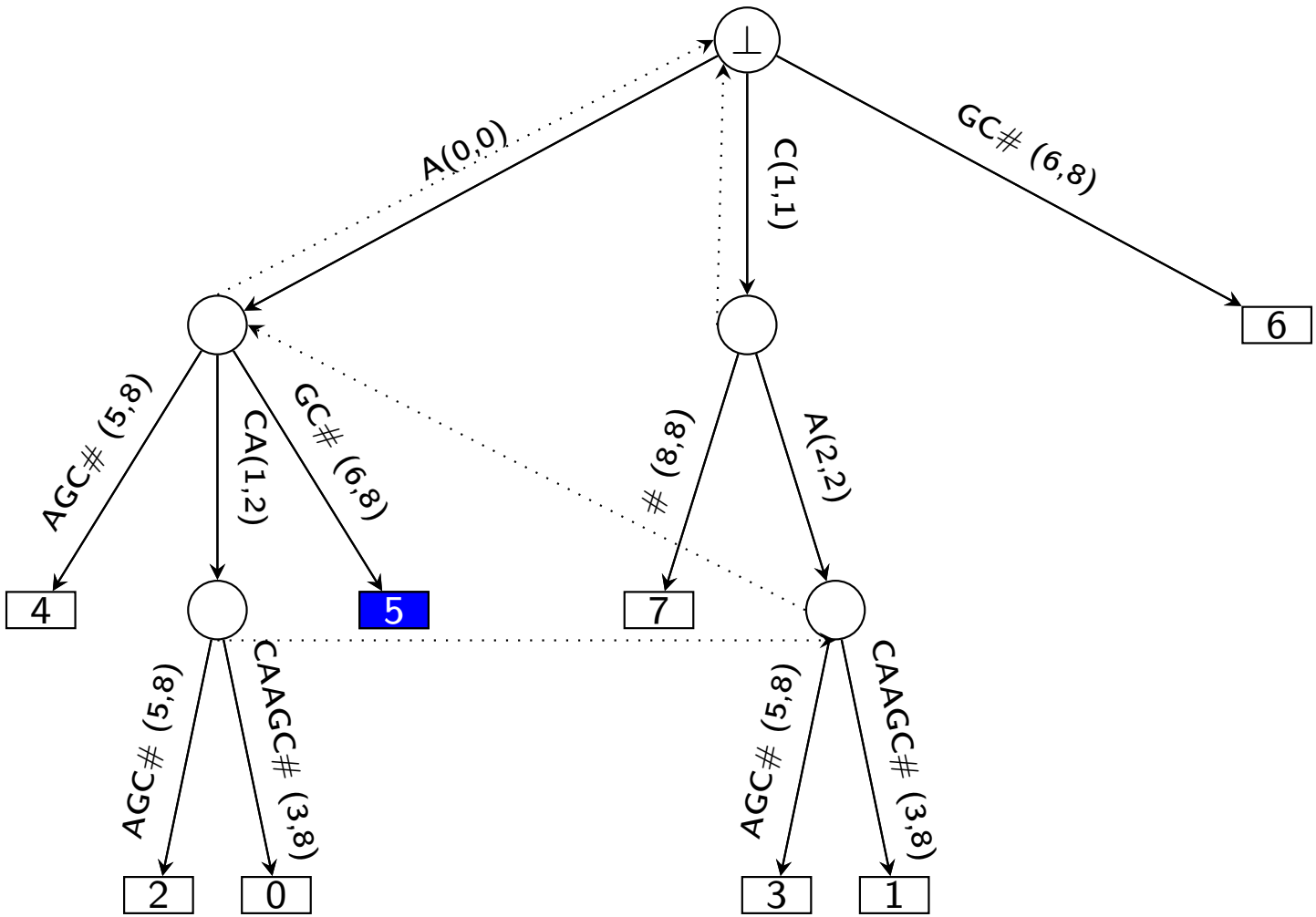
Suffix tree of $S =$ A C A C A A G C # (0 1 2 3 4 5 6 7 8)



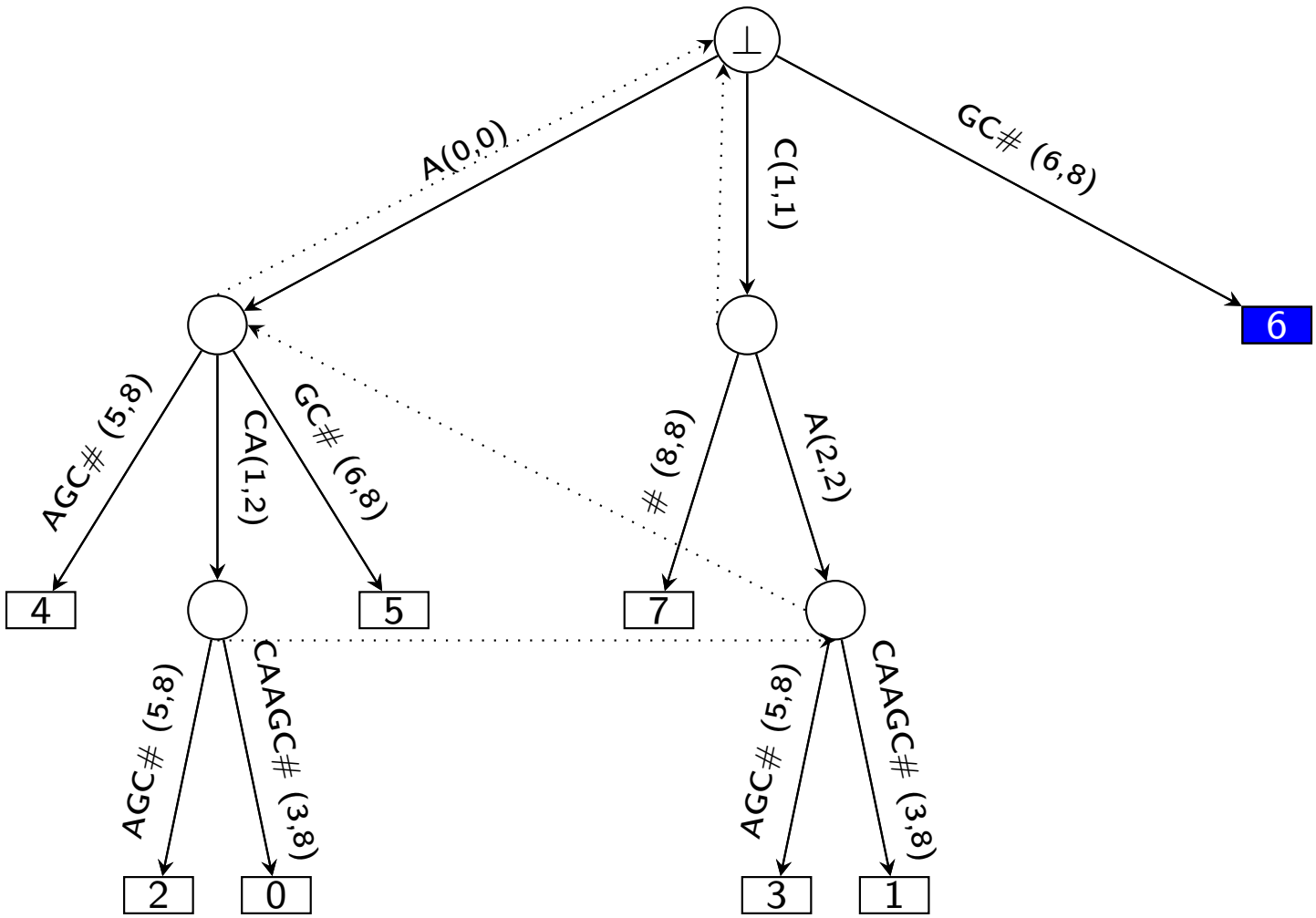
Suffix tree of $S =$ A C A C A A G C # (0 1 2 3 4 5 6 7 8)



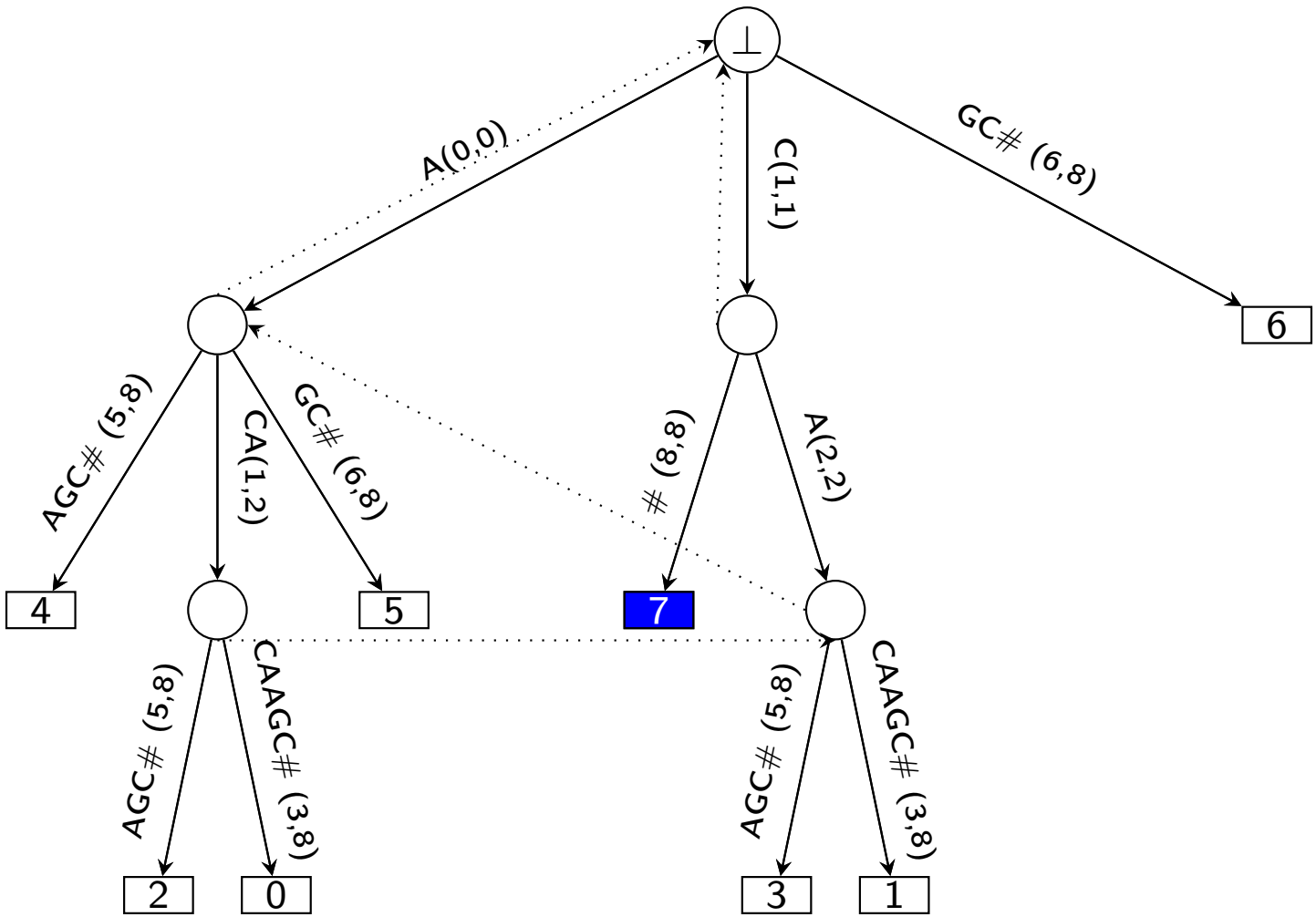
Suffix tree of $S =$ A C A C A A G C # (0 1 2 3 4 5 6 7 8)



Suffix tree of $S =$ A C A C A A G C # (0 1 2 3 4 5 6 7 8)

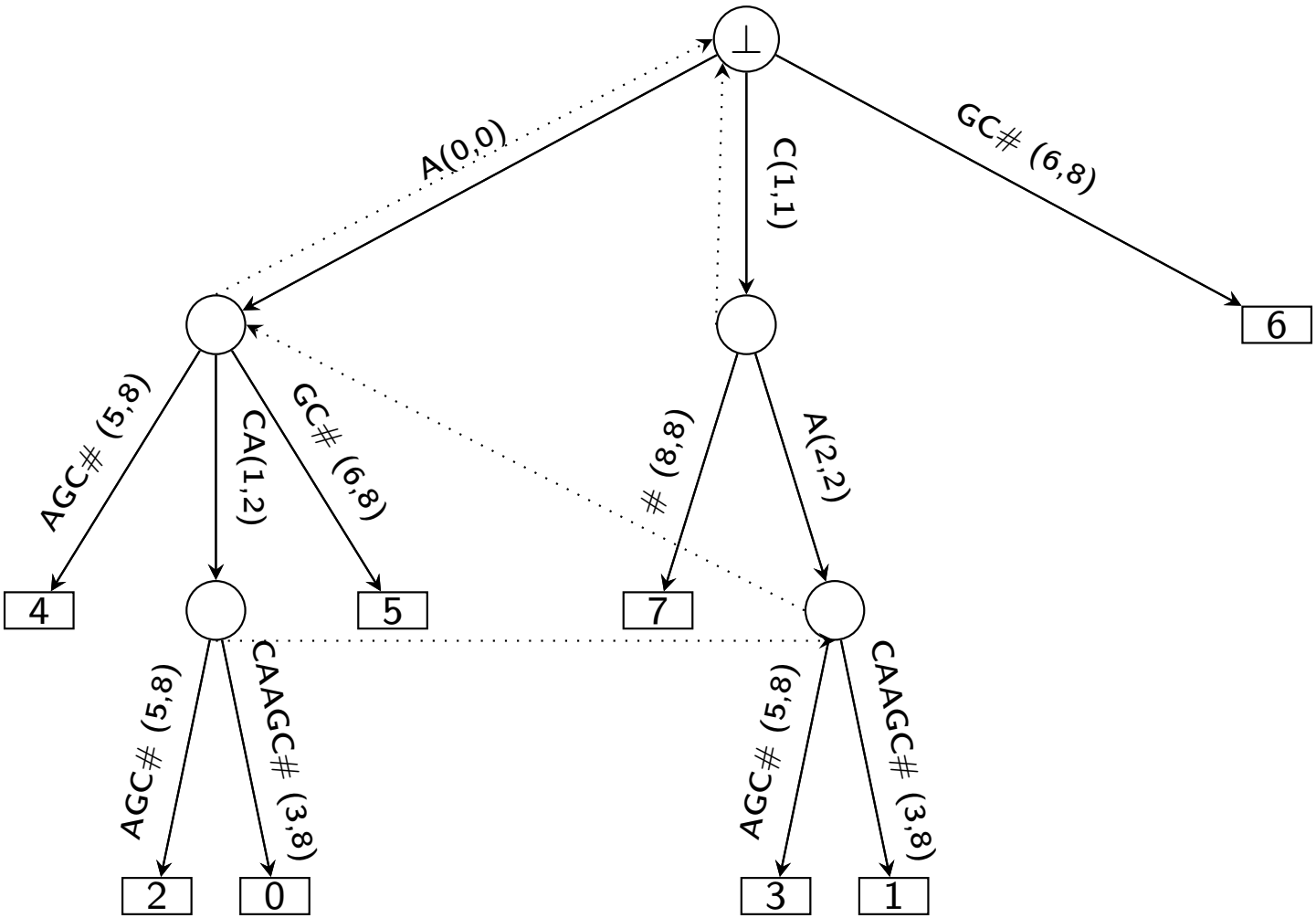


Suffix tree of $S =$ A C A C A A G C # (0 1 2 3 4 5 6 7 8)



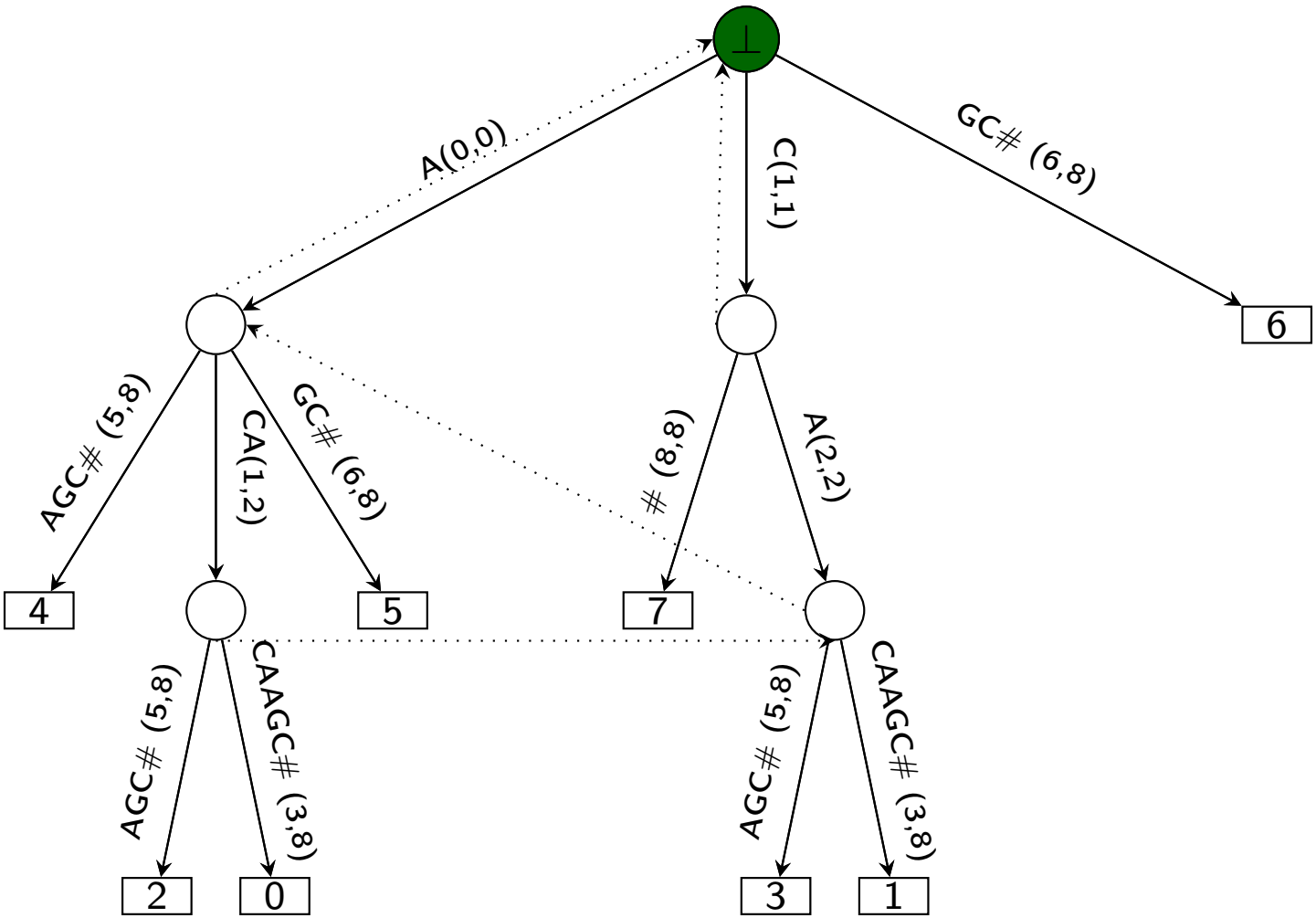
Suffix tree of $S =$ A C A C A A G C # (0 1 2 3 4 5 6 7 8)

AAA,AAC,CACAC,CAG,CC,CG,GA,GCA,GG



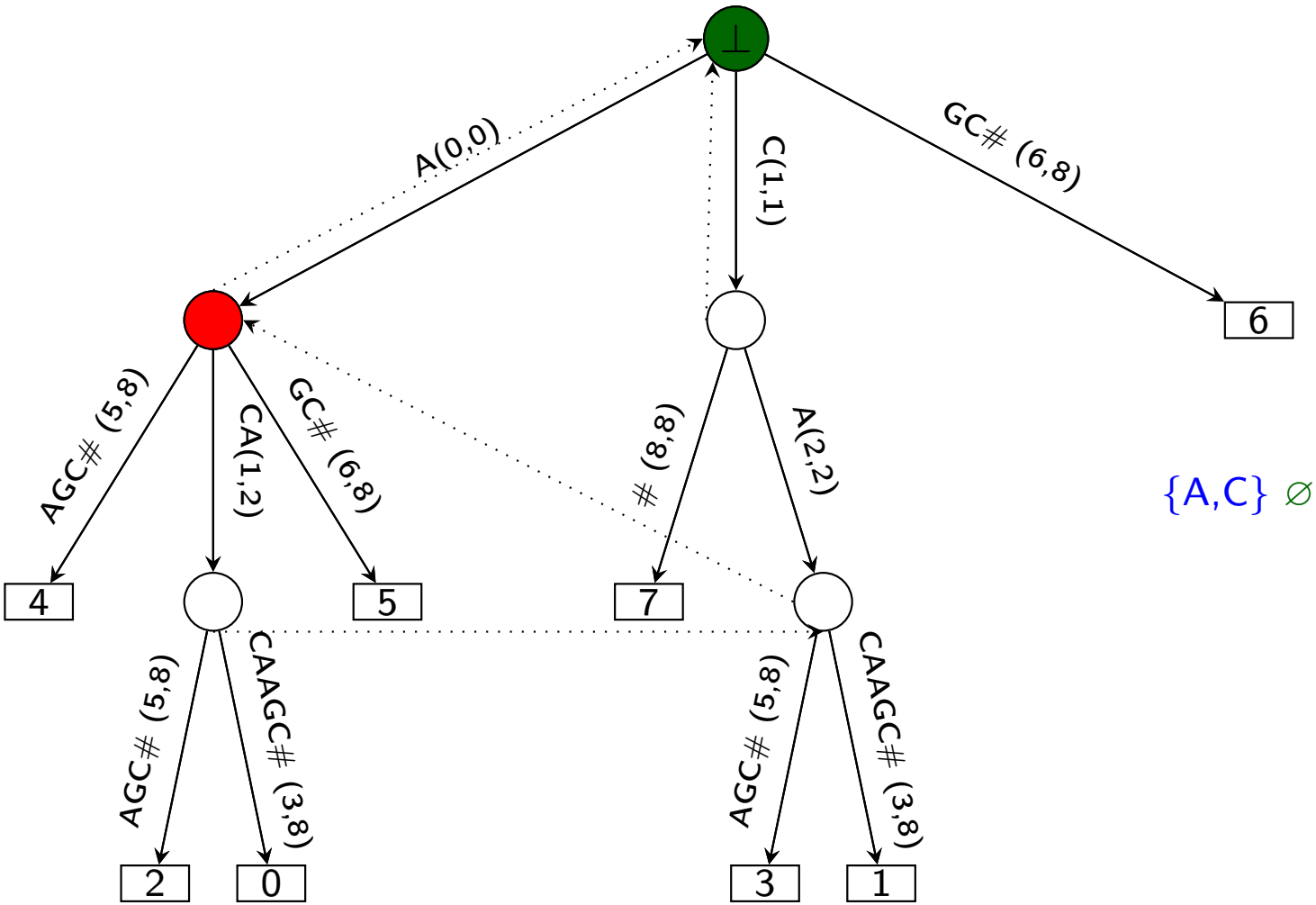
Suffix tree of S = 0 1 2 3 4 5 6 7 8
A C A C A A G C #

AAA,AAC,CACAC,CAG,CC,CG,GA,GCA,GG



Suffix tree of S = 0 1 2 3 4 5 6 7 8
A C A C A A G C #

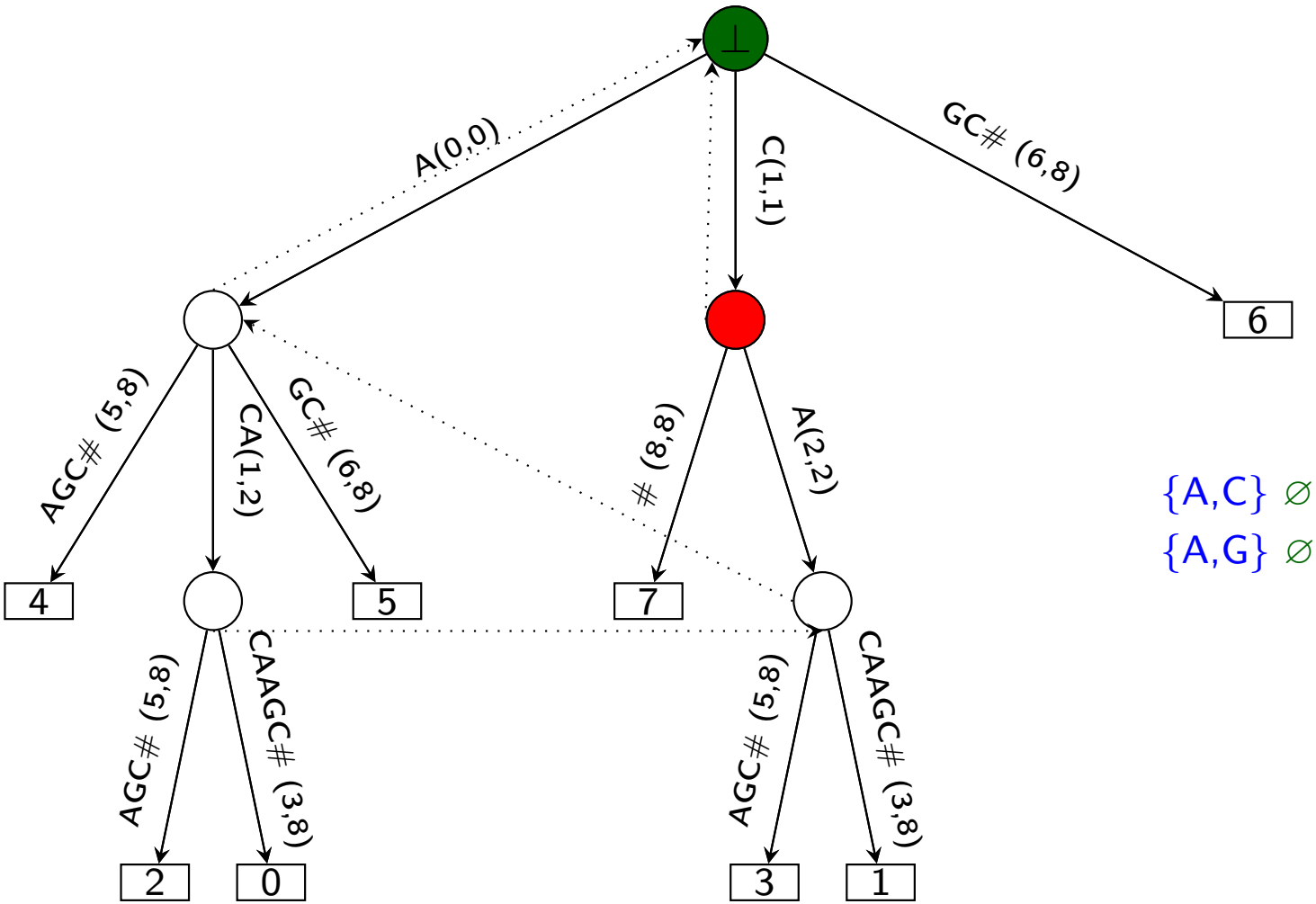
AAA,AAC,CACAC,CAG,CC,CG,GA,GCA,GG



{A,C} ∅ A

Suffix tree of S = A C A C A A G C#
0 1 2 3 4 5 6 7 8

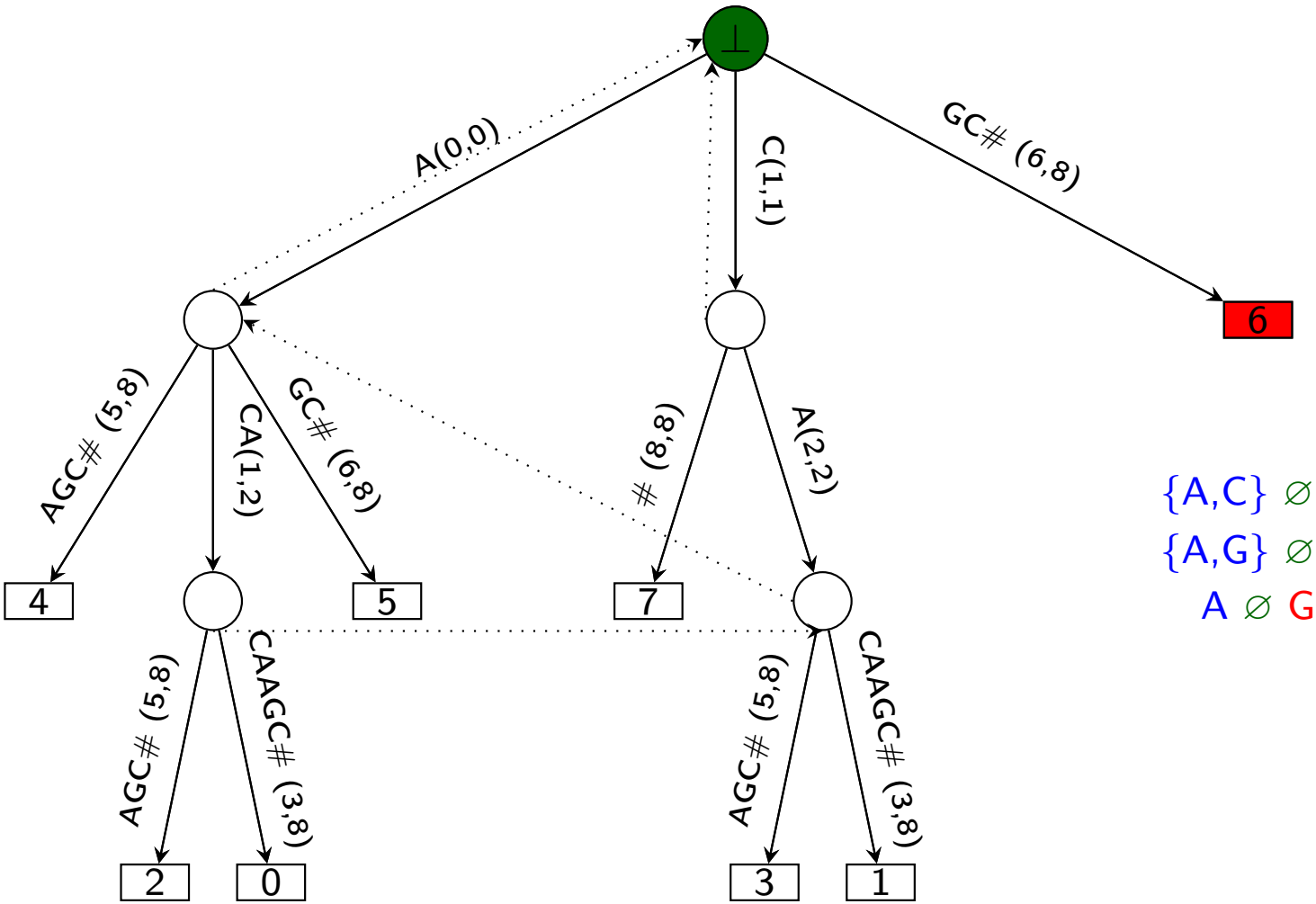
AAA,AAC,CACAC,CAG,CC,CG,GA,GCA,GG



{A,C} ⌀ A
{A,G} ⌀ C

Suffix tree of S = 0 1 2 3 4 5 6 7 8
A C A C A A G C #

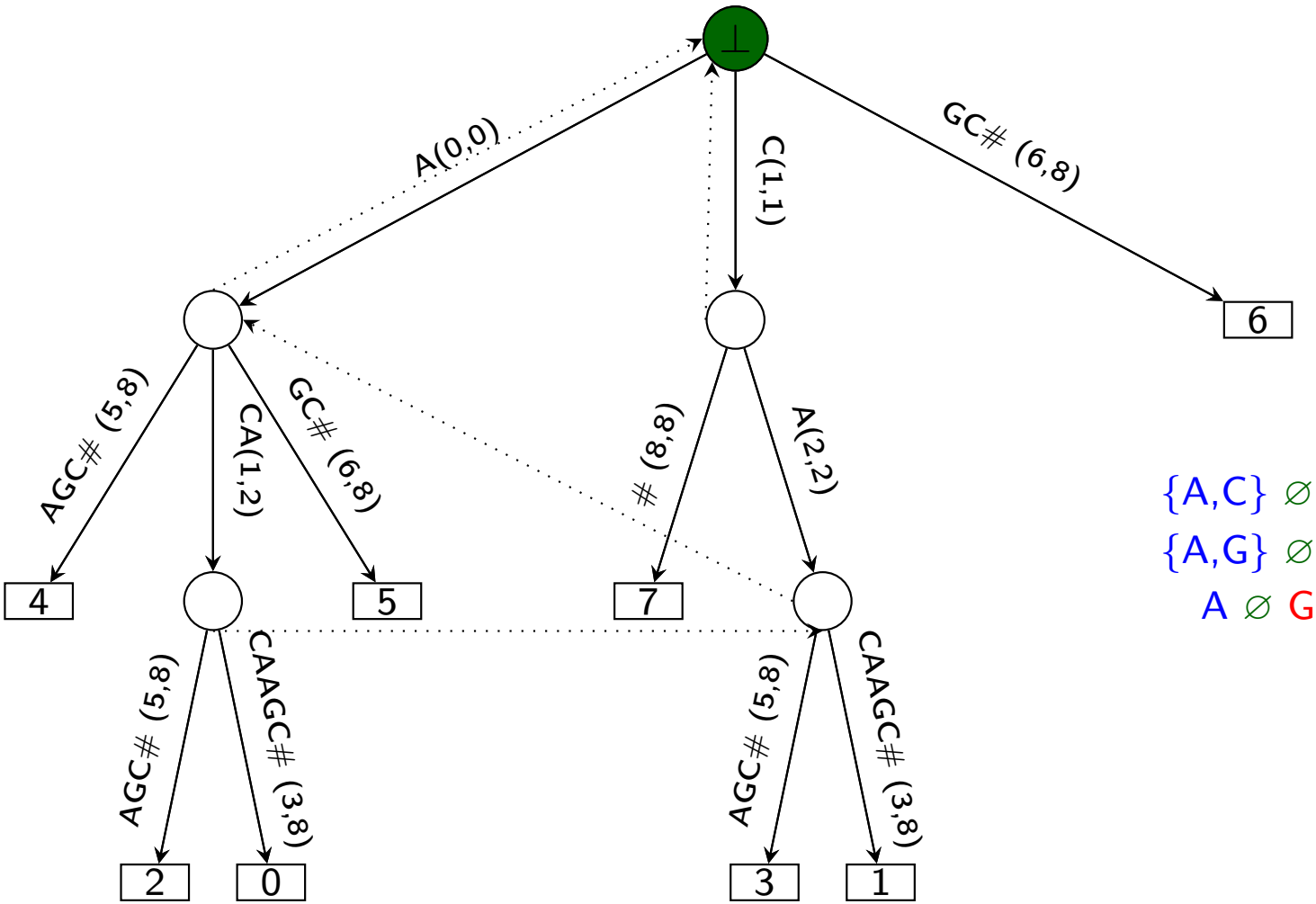
AAA,AAC,CACAC,CAG,CC,CG,GA,GCA,GG



{A,C} ⌀ A
{A,G} ⌀ C
A ⌀ G

Suffix tree of S = 0 1 2 3 4 5 6 7 8
A C A C A A G C #

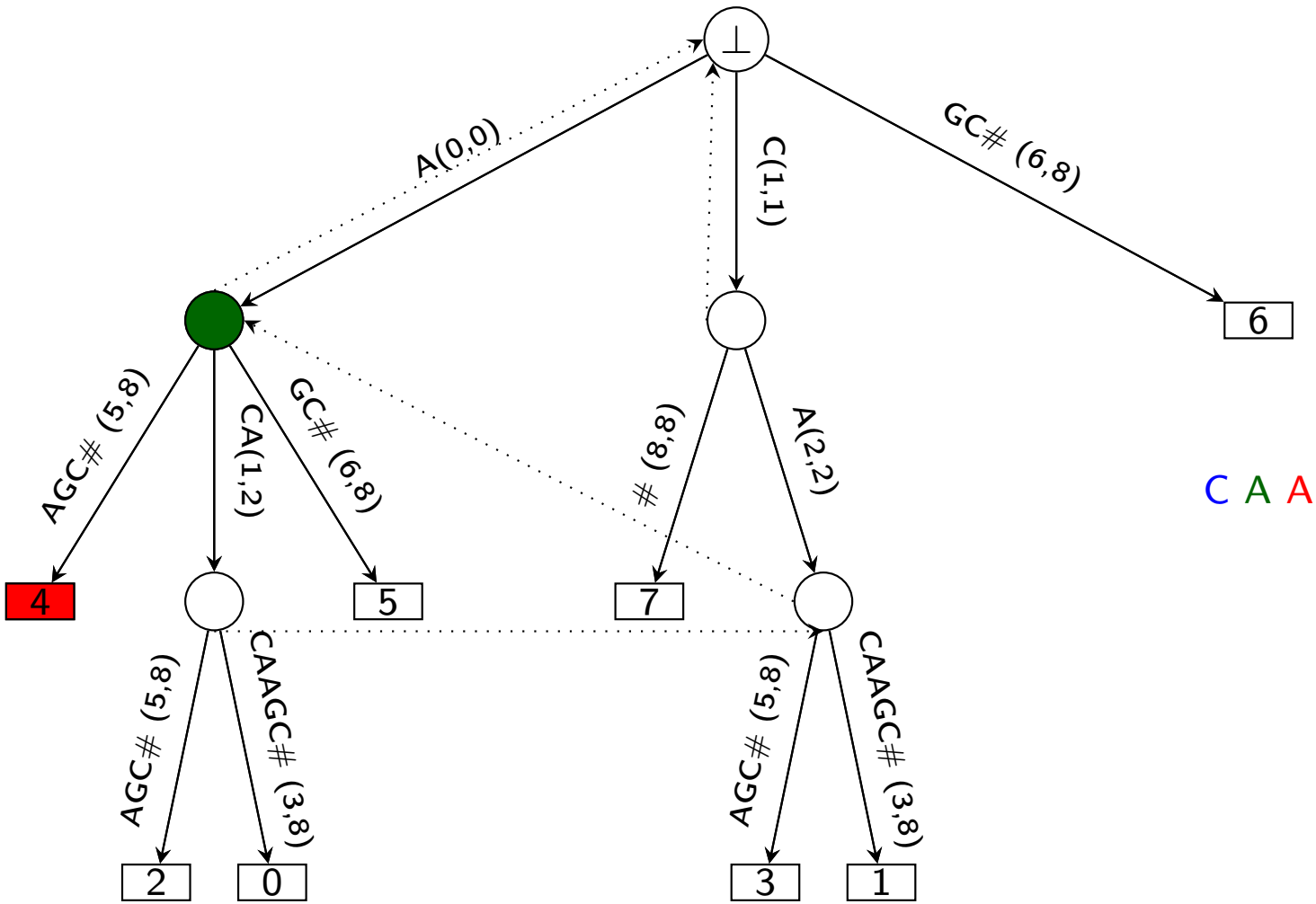
AAA,AAC,CACAC,CAG,CC,CG,GA,GCA,GG



{A,C} ∅ A
{A,G} ∅ C
A ∅ G

Suffix tree of S = 0 1 2 3 4 5 6 7 8
A C A C A A G C #

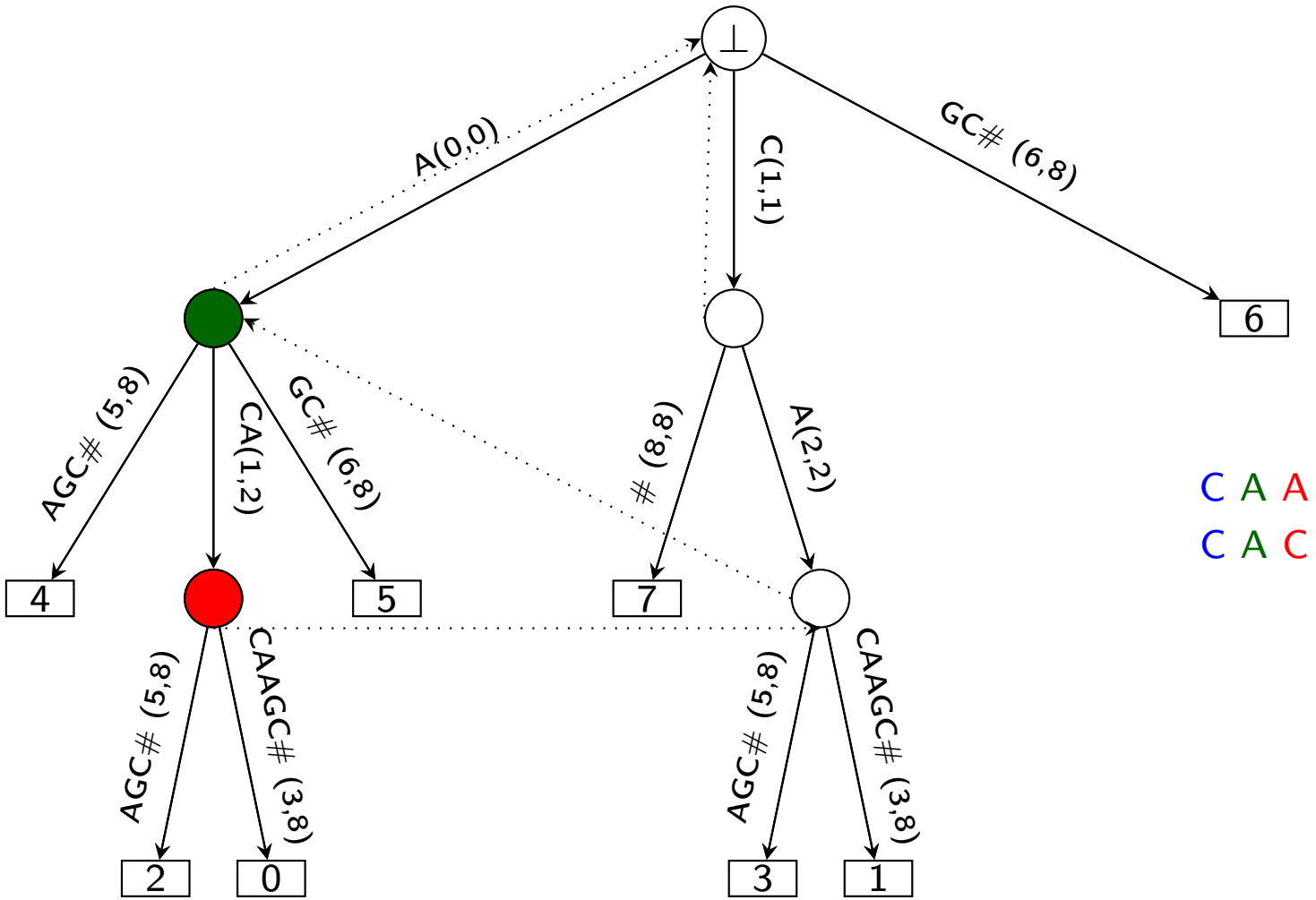
AAA, AAC, CACAC, CAG, CC, CG, GA, GCA, GG

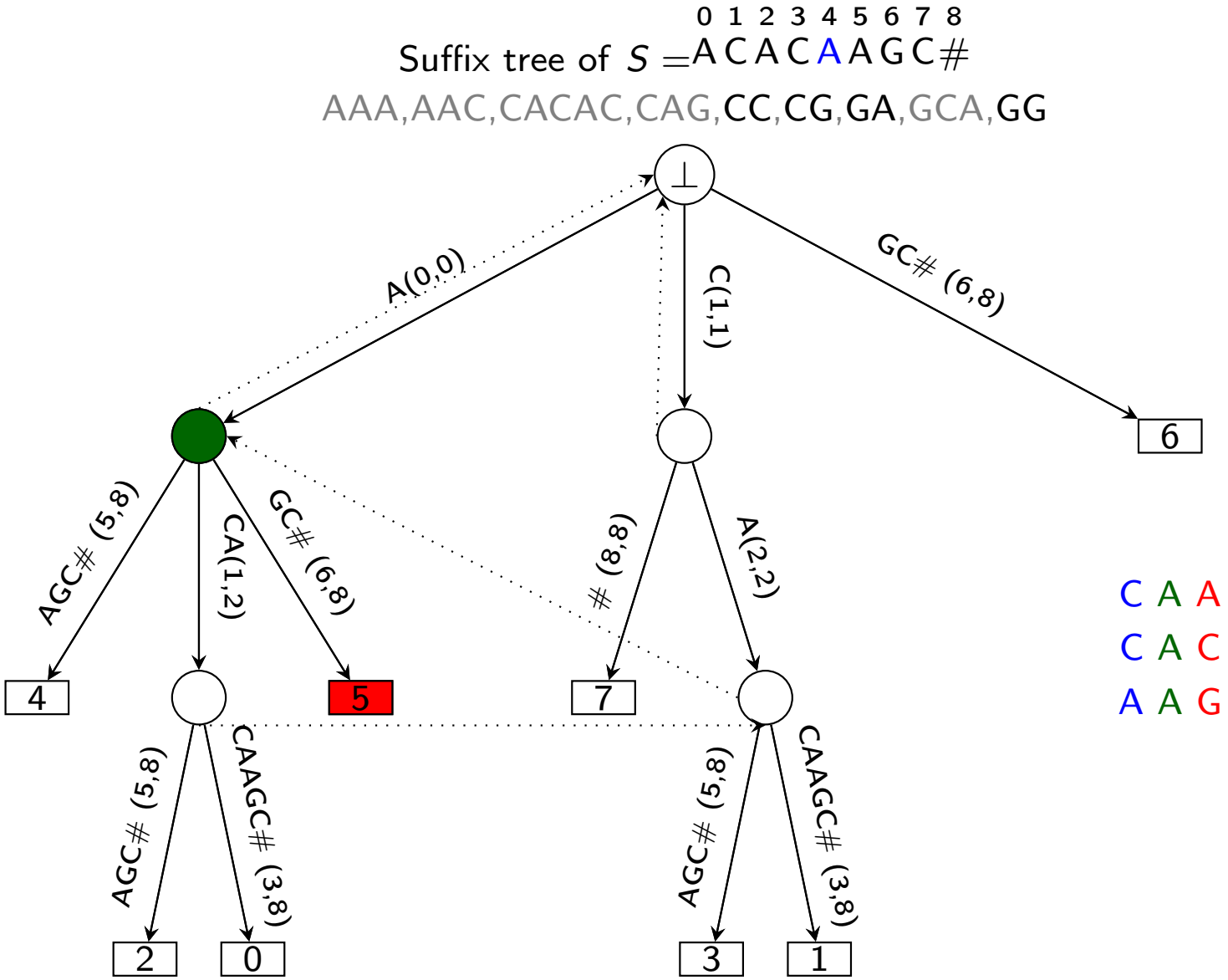


C A A

Suffix tree of S = 0 1 2 3 4 5 6 7 8
A C A C A A G C #

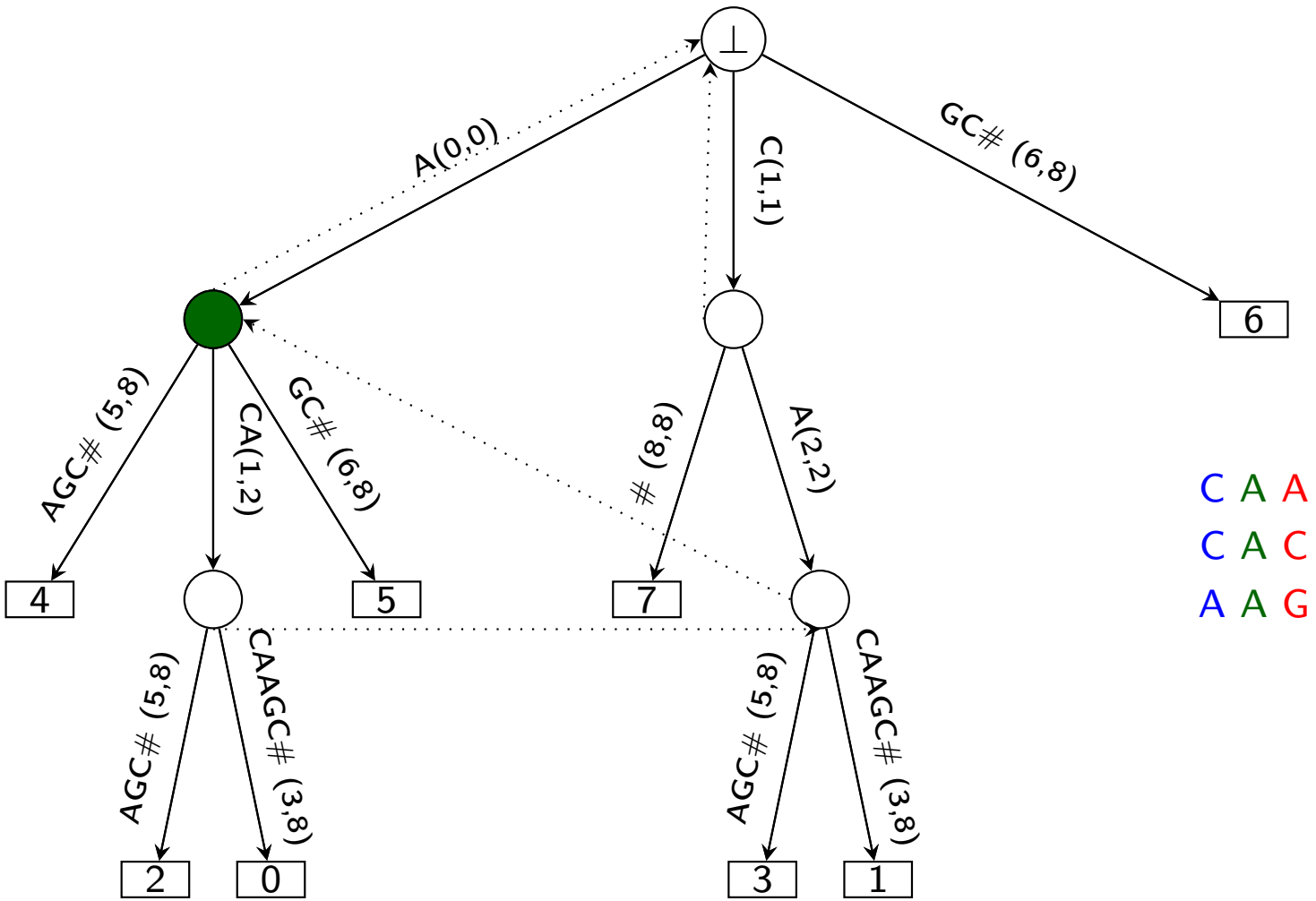
AAA, AAC, CACAC, CAG, CC, CG, GA, GCA, GG



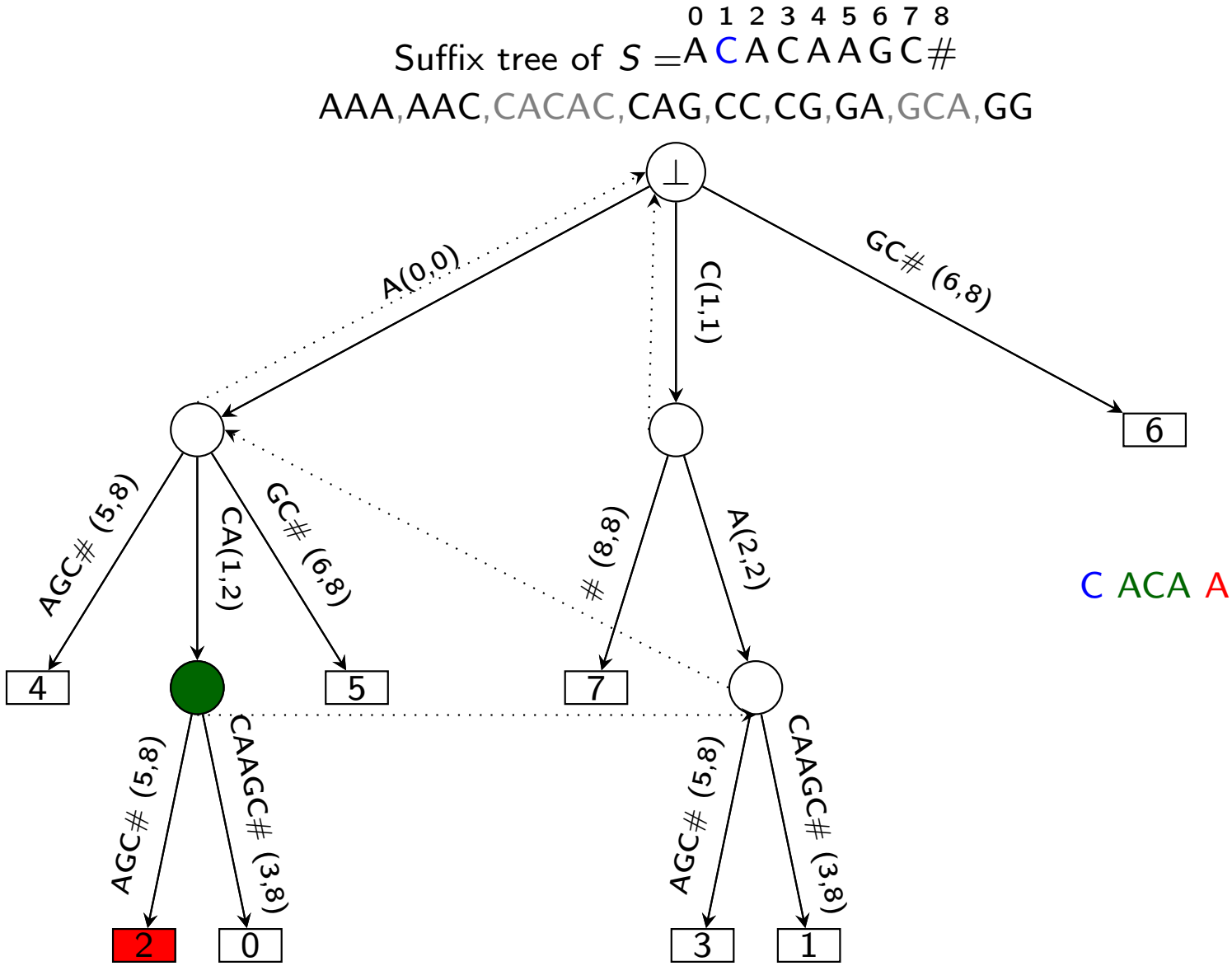

0 1 2 3 4 5 6 7 8
Suffix tree of S = A C A C A A G C #
AAA,AAC,CACAC,CAG,CC,CG,GA,GCA,GG
⊥
A(0,0)
C(1,1)
GC# (6,8)
6
AGC# (5,8)
CA(1,2)
GC# (6,8)
# (8,8)
A(2,2)
4
5
7
AGC# (5,8)
CAAGC# (3,8)
AGC# (5,8)
CAAGC# (3,8)
2
0
3
1
C A A
C A C
A A G

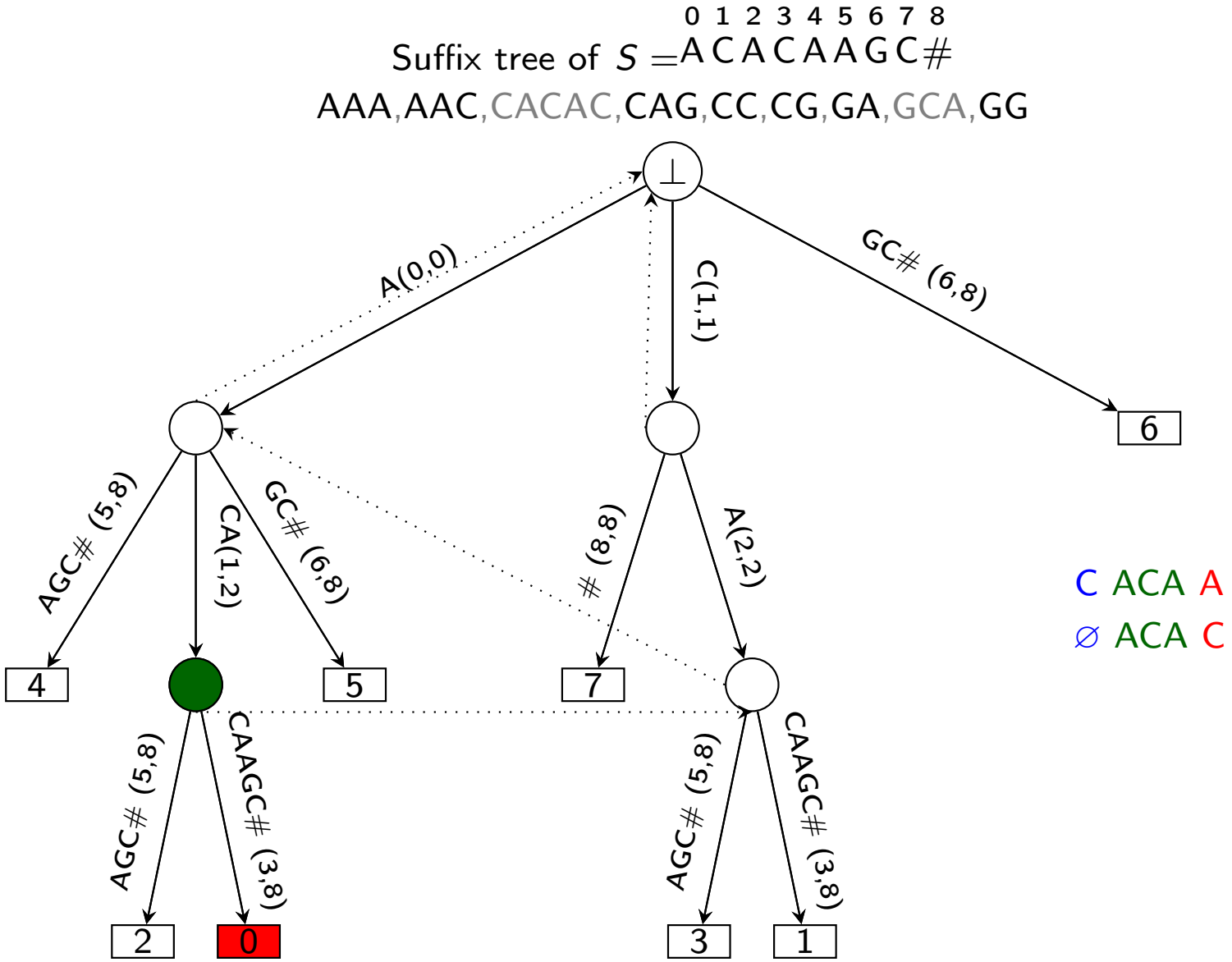
Suffix tree of S = A C A C A A G C #
0 1 2 3 4 5 6 7 8

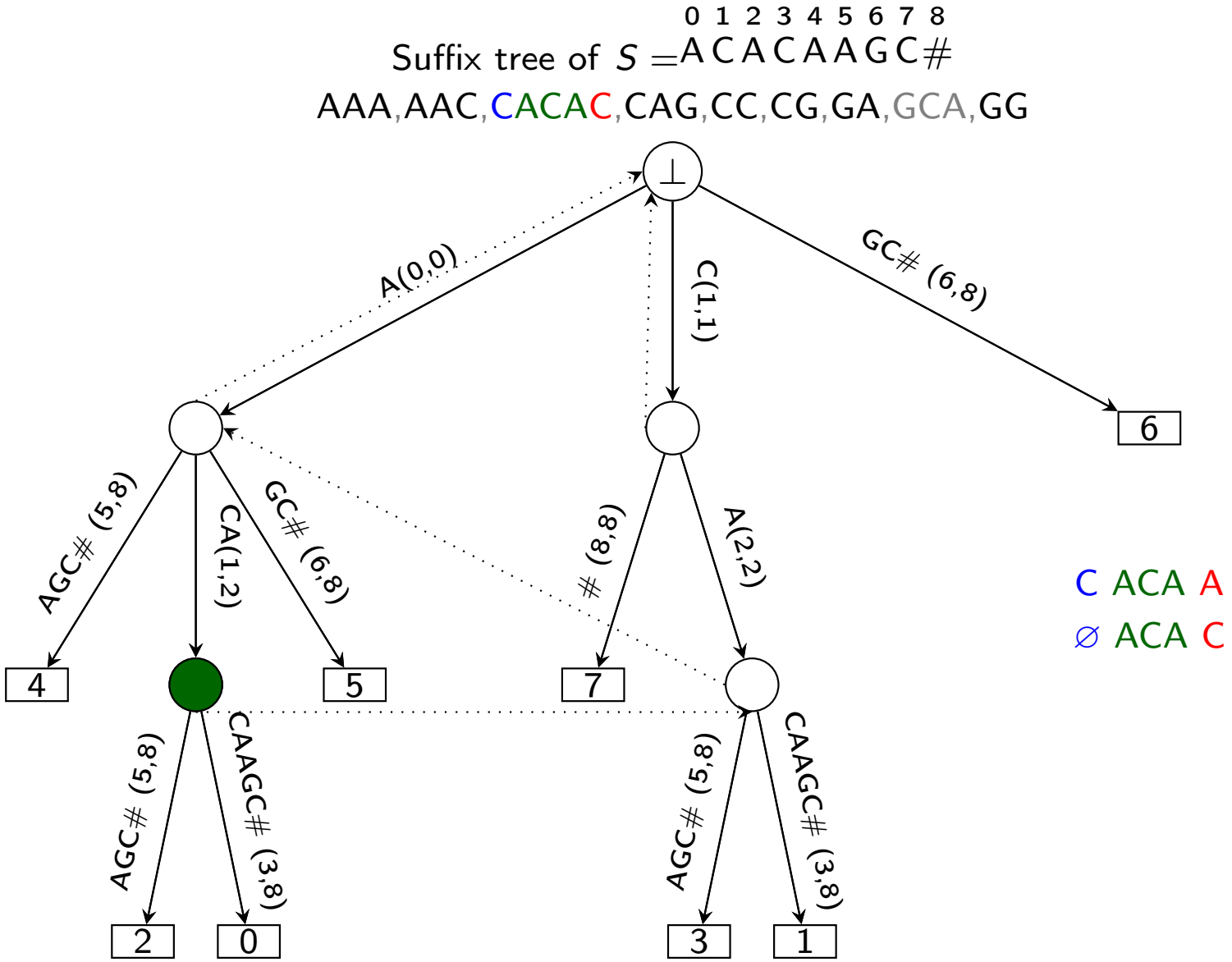
AAA, AAC, CACAC, CAG, CC, CG, GA, GCA, GG



C A A
C A C
A A G

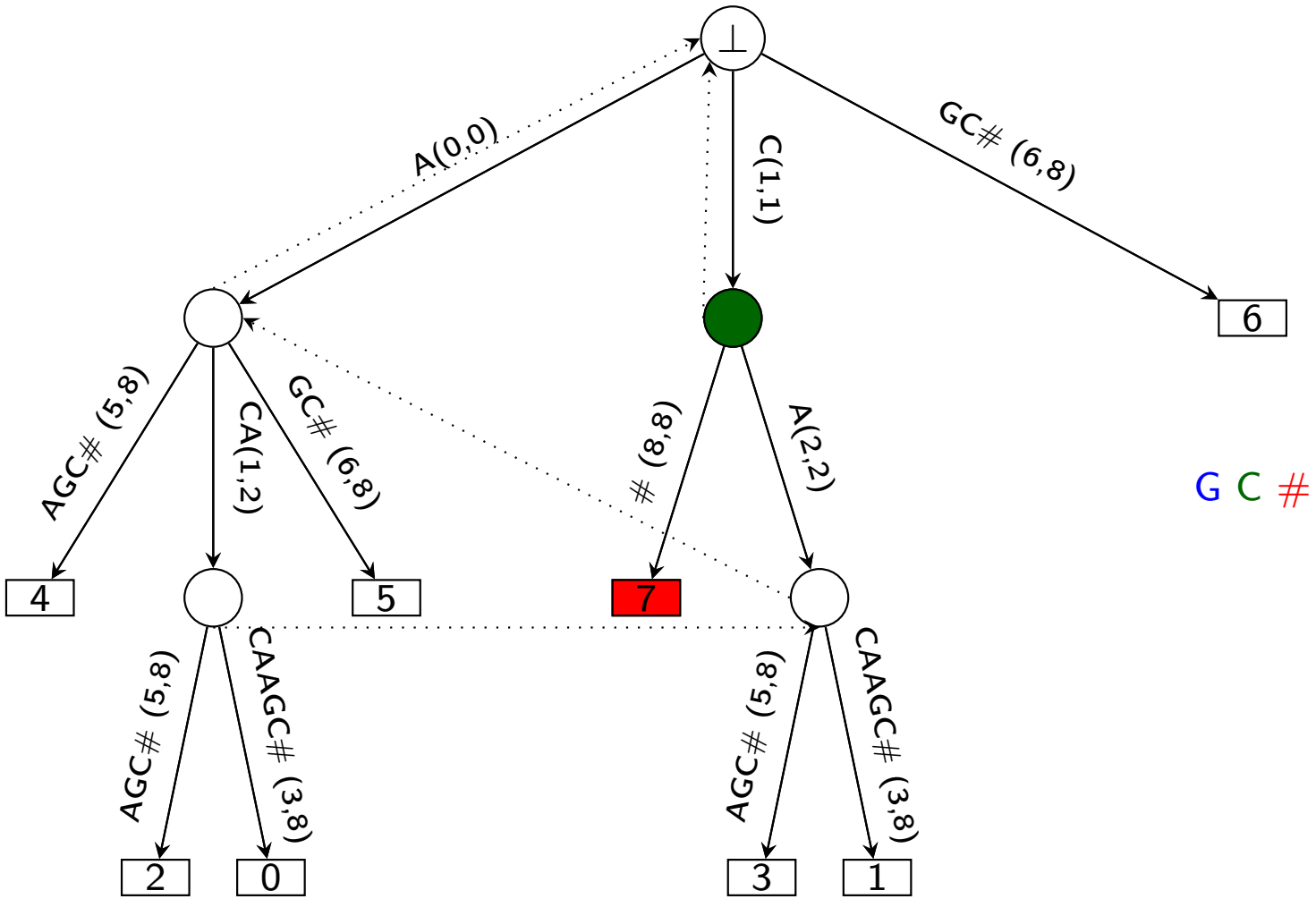

0 1 2 3 4 5 6 7 8
Suffix tree of S = A C A C A A G C #
AAA, AAC, CACAC, CAG, CC, CG, GA, GCA, GG
⊥
A(0,0)
C(1,1)
GC# (6,8)
6
AGC# (5,8)
CA(1,2)
GC# (6,8)
# (8,8)
A(2,2)
4
5
7
AGC# (5,8)
CAAGC# (3,8)
AGC# (5,8)
CAAGC# (3,8)
2
0
3
1
C ACA A


0 1 2 3 4 5 6 7 8
Suffix tree of S = A C A C A A G C #
AAA, AAC, CACAC, CAG, CC, CG, GA, GCA, GG
⊥
A(0,0)
C(1,1)
GC# (6,8)
6
AGC# (5,8)
CA(1,2)
GC# (6,8)
# (8,8)
A(2,2)
4
5
7
AGC# (5,8)
CAAGC# (3,8)
AGC# (5,8)
CAAGC# (3,8)
2
0
3
1
C ACA A
∅ ACA C


0 1 2 3 4 5 6 7 8
Suffix tree of S = A C A C A A G C #
AAA, AAC, CACAC, CAG, CC, CG, GA, GCA, GG
⊥
A(0,0)
C(1,1)
GC# (6,8)
6
AGC# (5,8)
CA(1,2)
GC# (6,8)
# (8,8)
A(2,2)
4
5
7
AGC# (5,8)
CAAGC# (3,8)
AGC# (5,8)
CAAGC# (3,8)
2
0
3
1
C ACA A
∅ ACA C

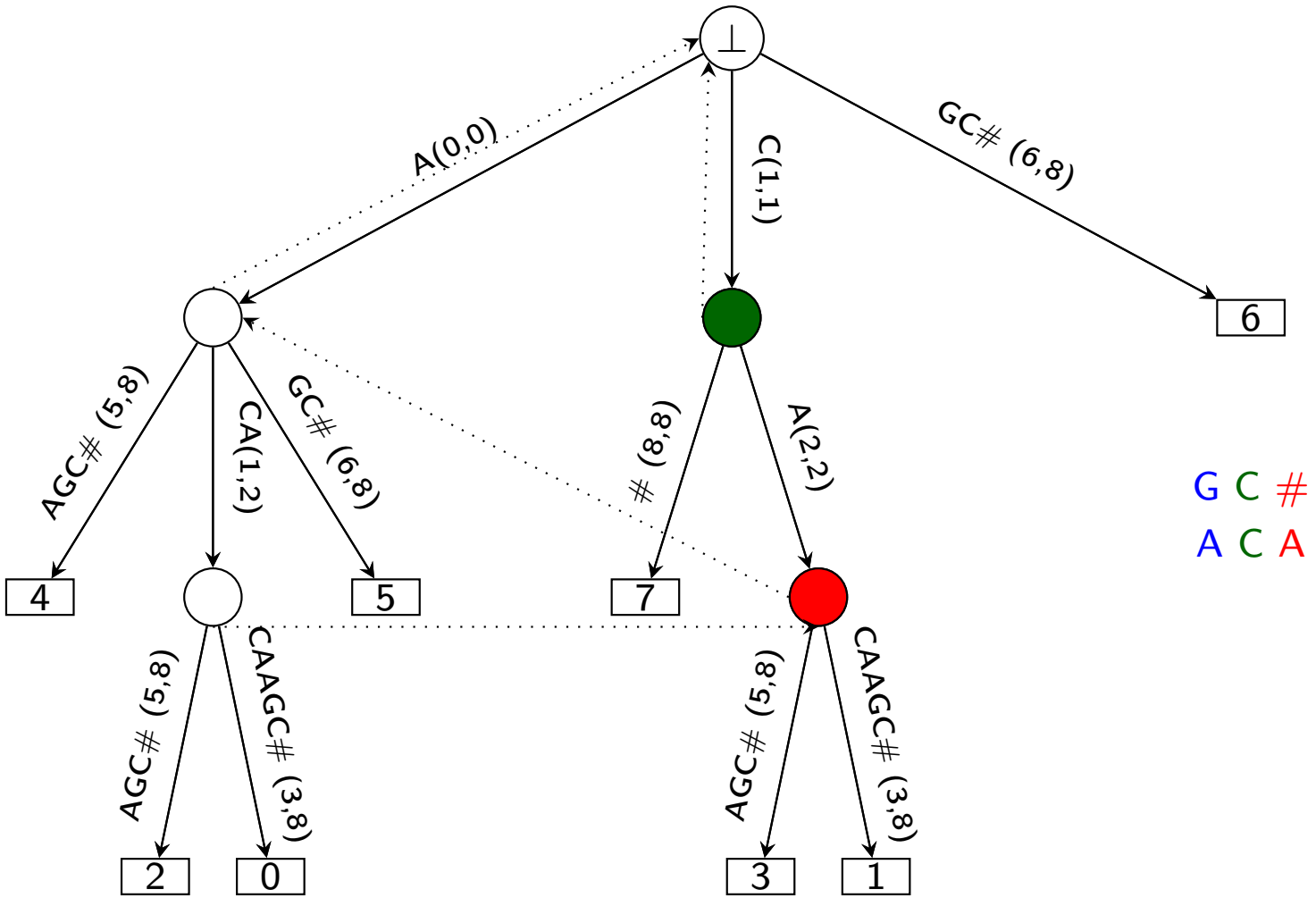
Suffix tree of $S =$ 0 1 2 3 4 5 6 7 8
A C A C A A G C #

AAA, AAC, CACAC, CAG, CC, CG, GA, GCA, GG



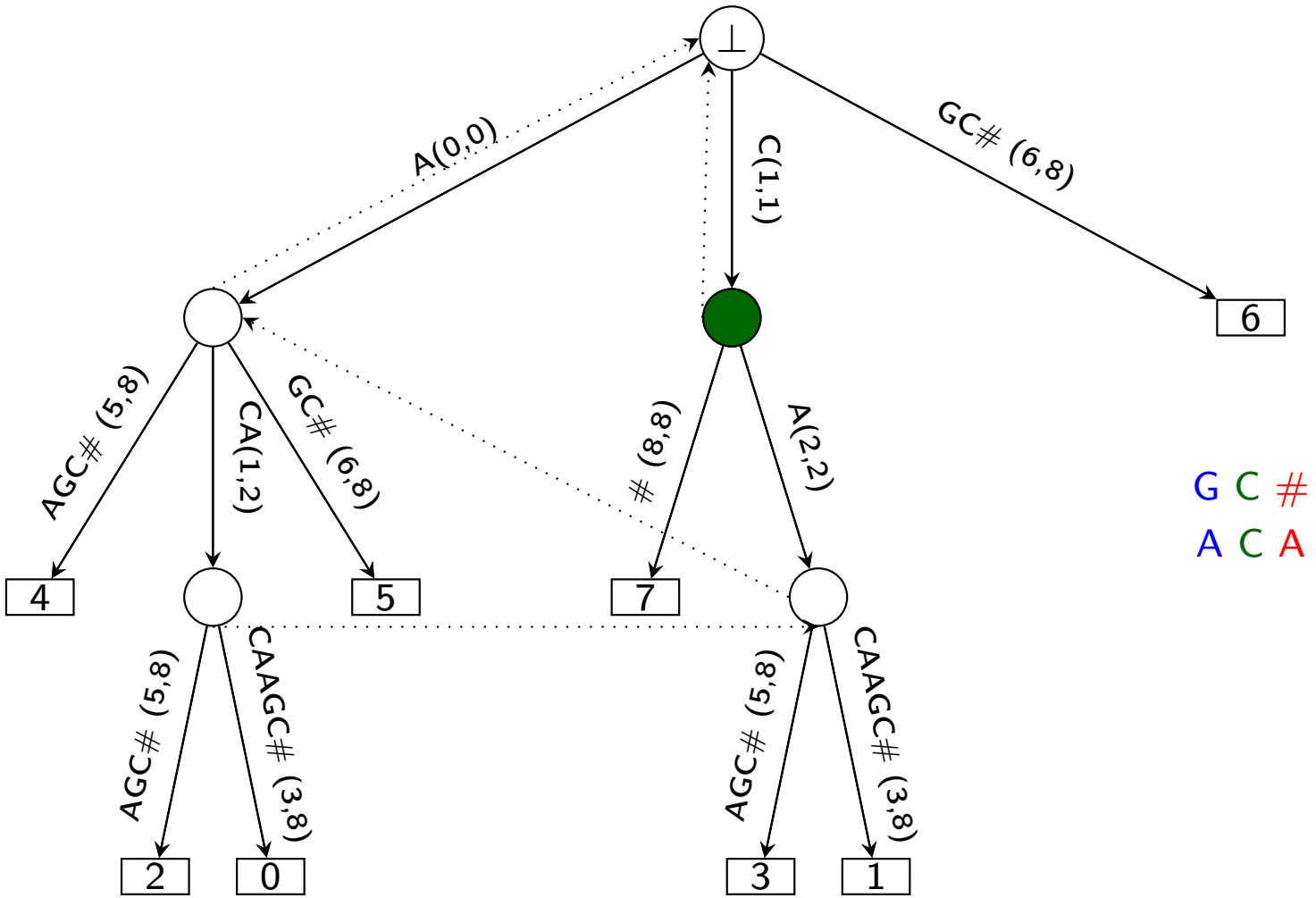
Suffix tree of $S =$ 0 1 2 3 4 5 6 7 8
A C A C A A G C #

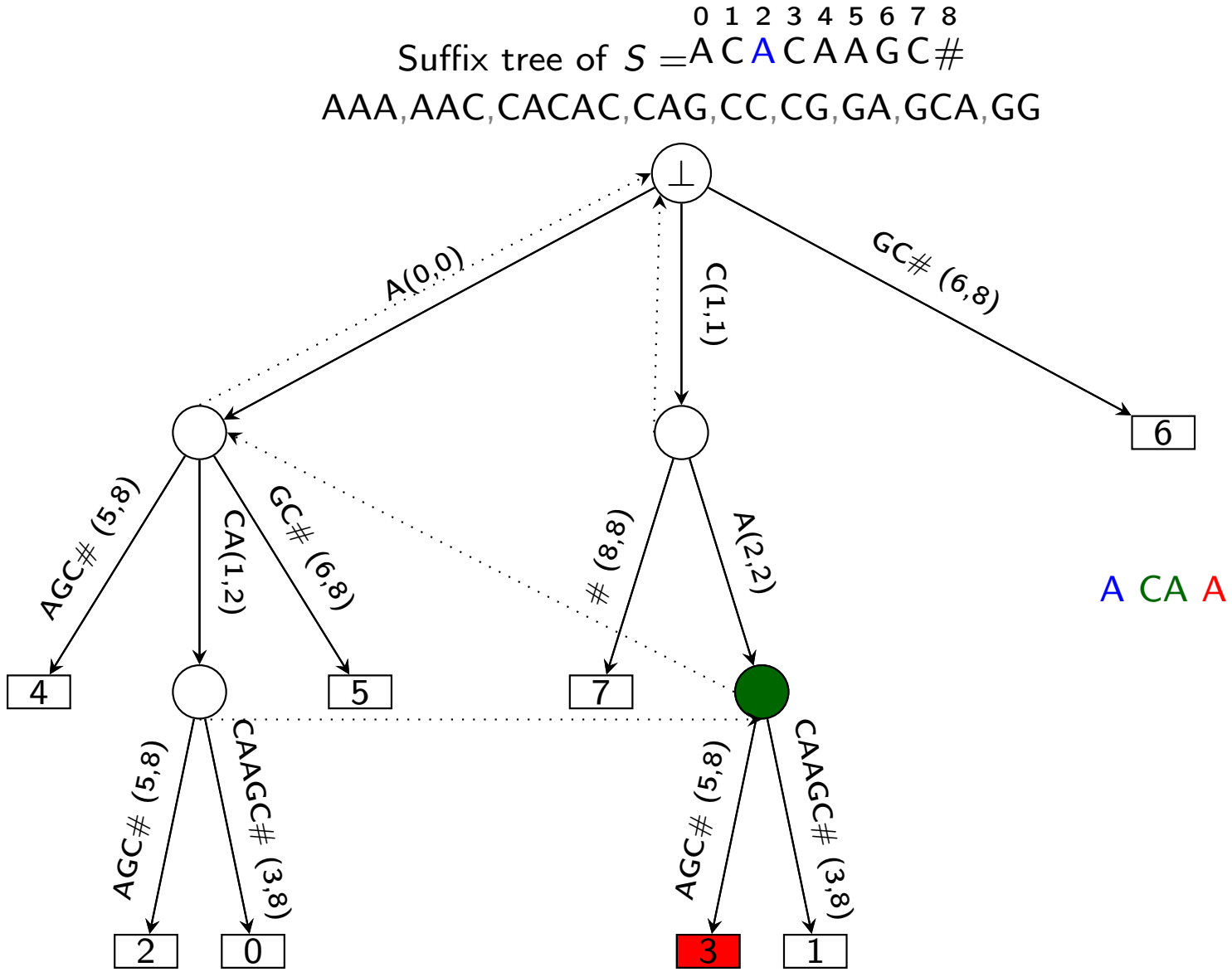
AAA, AAC, CACAC, CAG, CC, CG, GA, GCA, GG

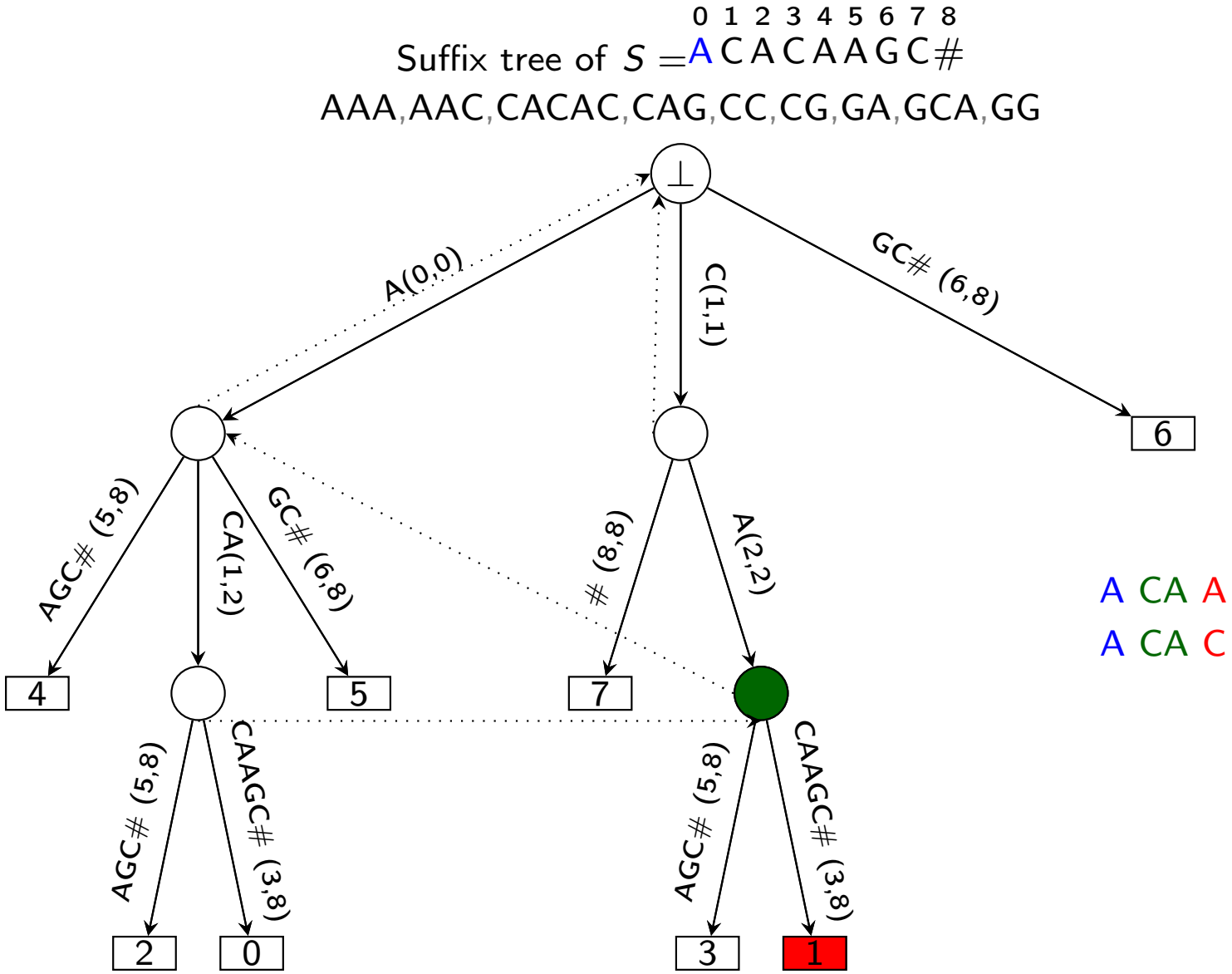


Suffix tree of $S =$ 0 1 2 3 4 5 6 7 8
A C A C A A G C #

AAA, AAC, CACAC, CAG, CC, CG, GA, GCA, GG

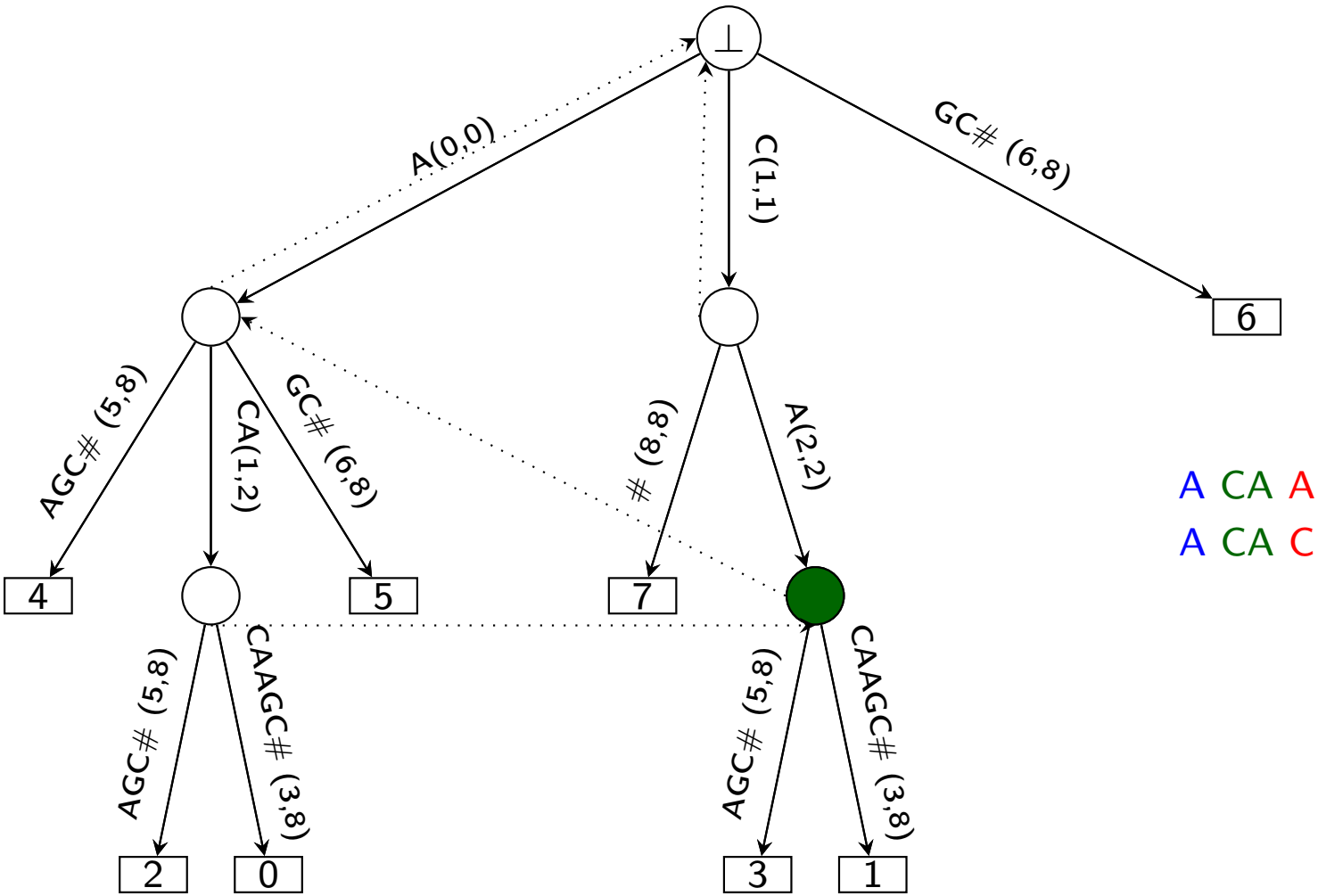



0 1 2 3 4 5 6 7 8
Suffix tree of S = A C A C A A G C #
AAA, AAC, CACAC, CAG, CC, CG, GA, GCA, GG
⊥
A(0,0)
C(1,1)
GC# (6,8)
6
AGC# (5,8)
CA(1,2)
GC# (6,8)
# (8,8)
A(2,2)
4
5
7
AGC# (5,8)
CAAGC# (3,8)
AGC# (5,8)
CAAGC# (3,8)
2
0
3
1
A CA A


0 1 2 3 4 5 6 7 8
Suffix tree of S = A C A C A A G C #
AAA, AAC, CACAC, CAG, CC, CG, GA, GCA, GG
⊥
A(0,0)
C(1,1)
GC# (6,8)
6
AGC# (5,8)
CA(1,2)
GC# (6,8)
# (8,8)
A(2,2)
4
5
7
AGC# (5,8)
CAAGC# (3,8)
AGC# (5,8)
CAAGC# (3,8)
2
0
3
1
A CA A
A CA C

Suffix tree of $S =$ 0 1 2 3 4 5 6 7 8
A C A C A A G C #

AAA, AAC, CACAC, CAG, CC, CG, GA, GCA, GG



A CA A
A CA C

1. Minimal absent words

2. Minimal absent words over a sliding window
   - Problem definition
   - Ukkonen construction algorithm of the suffix tree
   - The suffix tree for a sliding window
   - Algorithm for maws in a sliding window

Minimal absent words on a sliding window

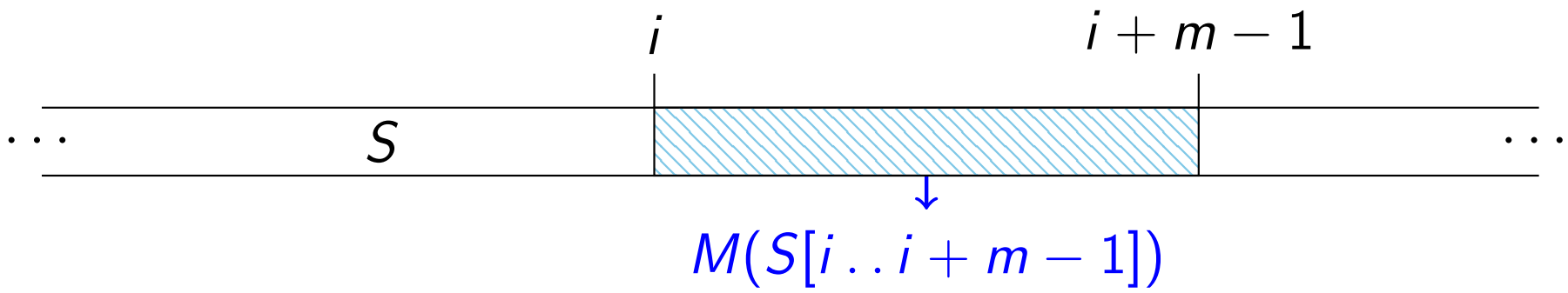
- Sequence S of size n, over a constant size alphabet,
- Sliding window of size m on S : $S[i \,..\, i + m - 1]$.

For all word x we denote by $M(x)$ its set of minimal absent words.

Minimal absent words on a sliding window

- Sequence S of size n, over a constant size alphabet,
- Sliding window of size m on S : $S[i \,..\, i+m-1]$.

For all word x we denote by $M(x)$ its set of minimal absent words.



Minimal absent words on a sliding window

Lemma

$\sum_{i=0}^{n-m} |M(y[i \,..\, i+m-1])|$ is upper bounded by $\mathcal{O}(nm)$.

Minimal absent words on a sliding window

Lemma

$\sum_{i=0}^{n-m} |M(y[i \,..\, i+m-1])|$ is upper bounded by $\mathcal{O}(nm)$.

$\rightarrow$ We can't output the set if minimal absent words for each factor of size m in time $\mathcal{O}(n)$.

Minimal absent words on a sliding window

Lemma

$\sum_{i=0}^{n-m} |M(y[i\,..\,i+m-1])|$ is upper bounded by $\mathcal{O}(nm)$.

$\rightarrow$ We can't output the set if minimal absent words for each factor of size m in time $\mathcal{O}(n)$.

Theorem

The upper bound of $\sum_{i=0}^{n-m-1} |M(y[i\,..\,i+m-1]) \triangle M(y[i+1\,..\,i+m])|$ is $\mathcal{O}(n)$.

Minimal absent words on a sliding window

Lemma

$\sum_{i=0}^{n-m} |M(y[i\,..\,i+m-1])|$ is upper bounded by $\mathcal{O}(nm)$.

$\rightarrow$ We can't output the set if minimal absent words for each factor of size m in time $\mathcal{O}(n)$.

Theorem

The upper bound of $\sum_{i=0}^{n-m-1} |M(y[i\,..\,i+m-1]) \triangle M(y[i+1\,..\,i+m])|$ is $\mathcal{O}(n)$.

$\rightarrow$ We need a dynamic structure to go from one set to another efficiently

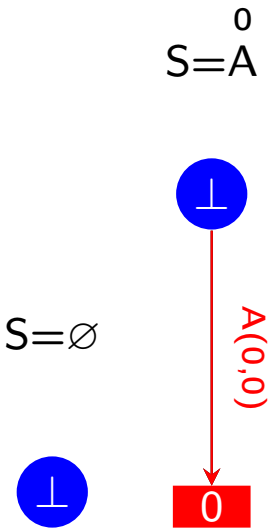
Dynamic construction of the suffix tree

- From left to right by Weiner 1973,
- From right to left by McCreight in 1976,
- Ukkonen algorithm in 1995 provides a more intuitive algorithm.

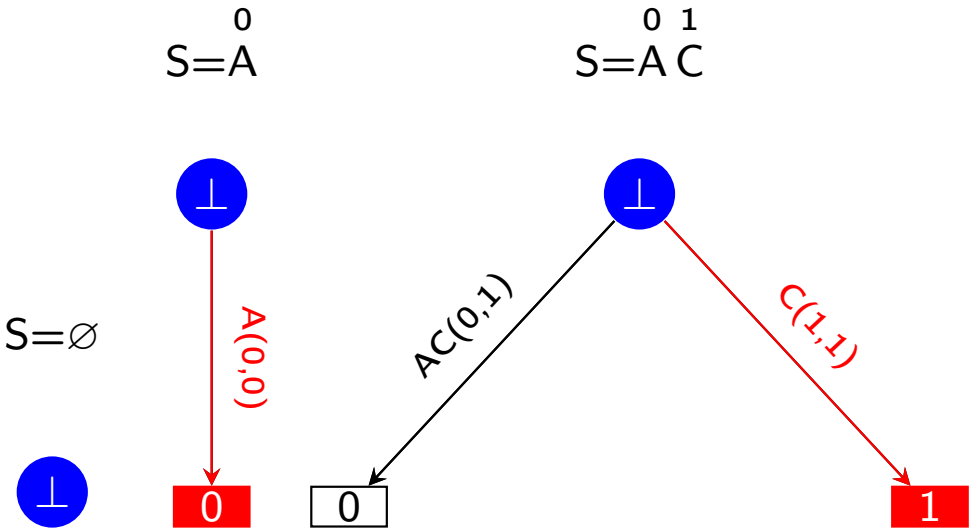
Ukkonen construction algorithm of the suffix tree

$S=\varnothing$

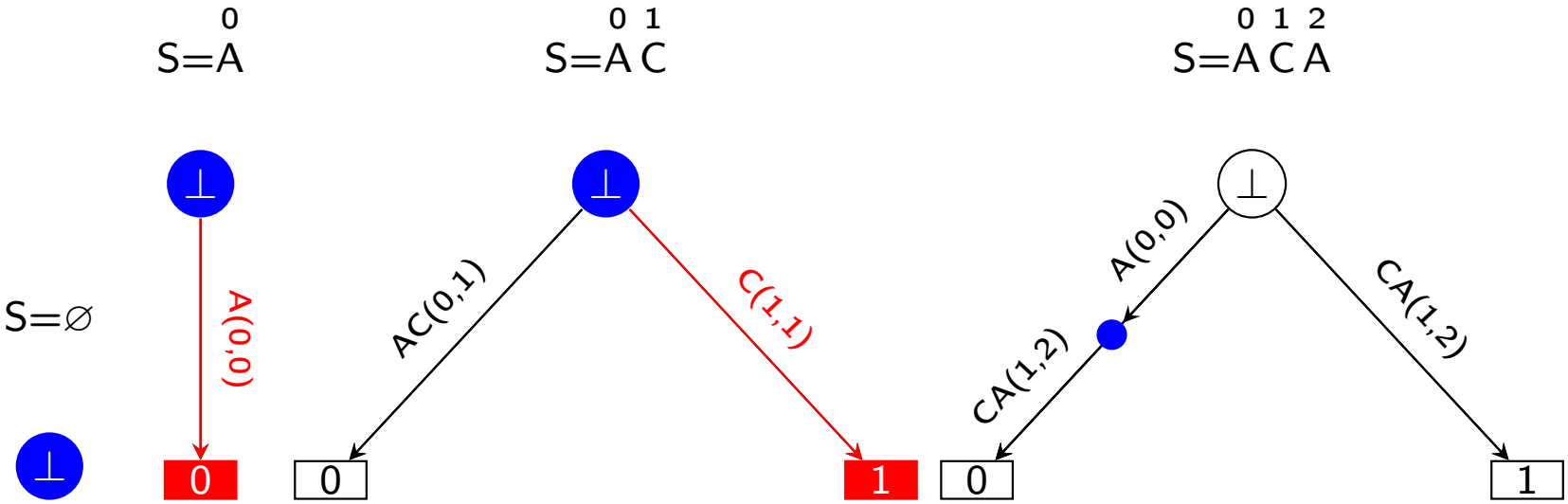
Ukkonen construction algorithm of the suffix tree

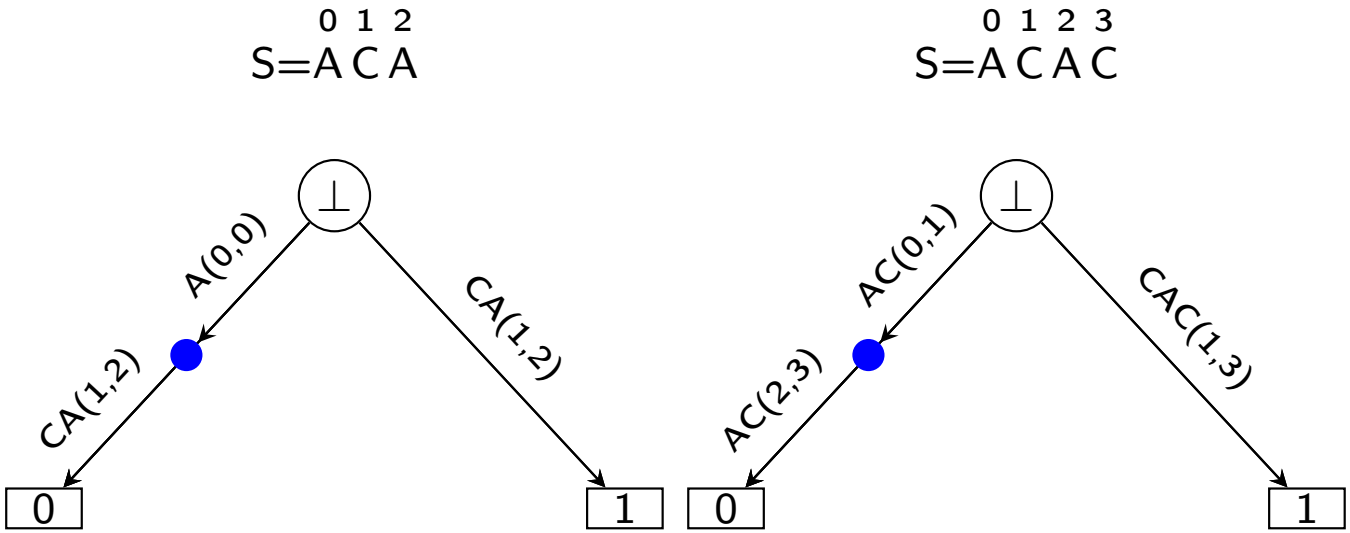


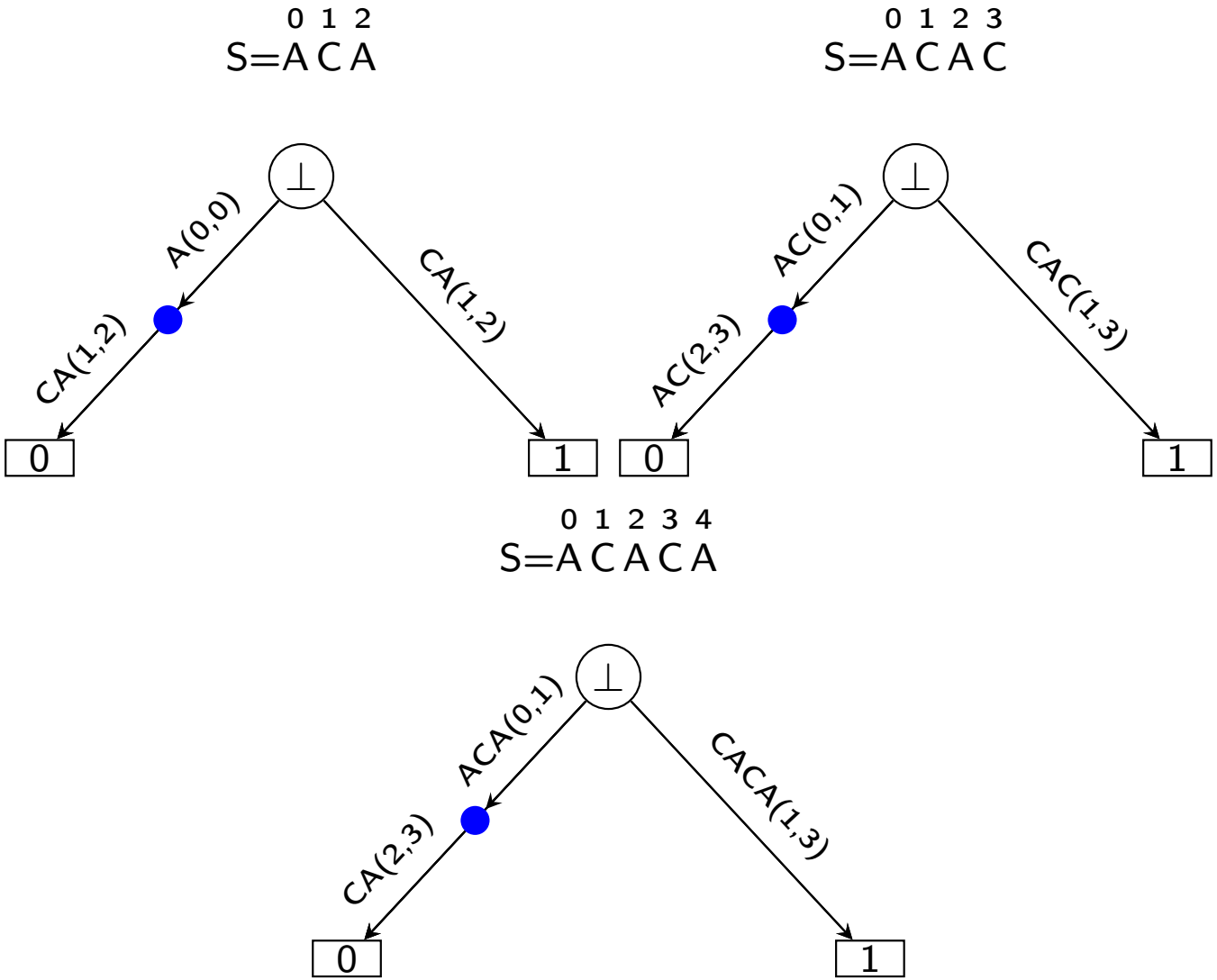
Ukkonen construction algorithm of the suffix tree

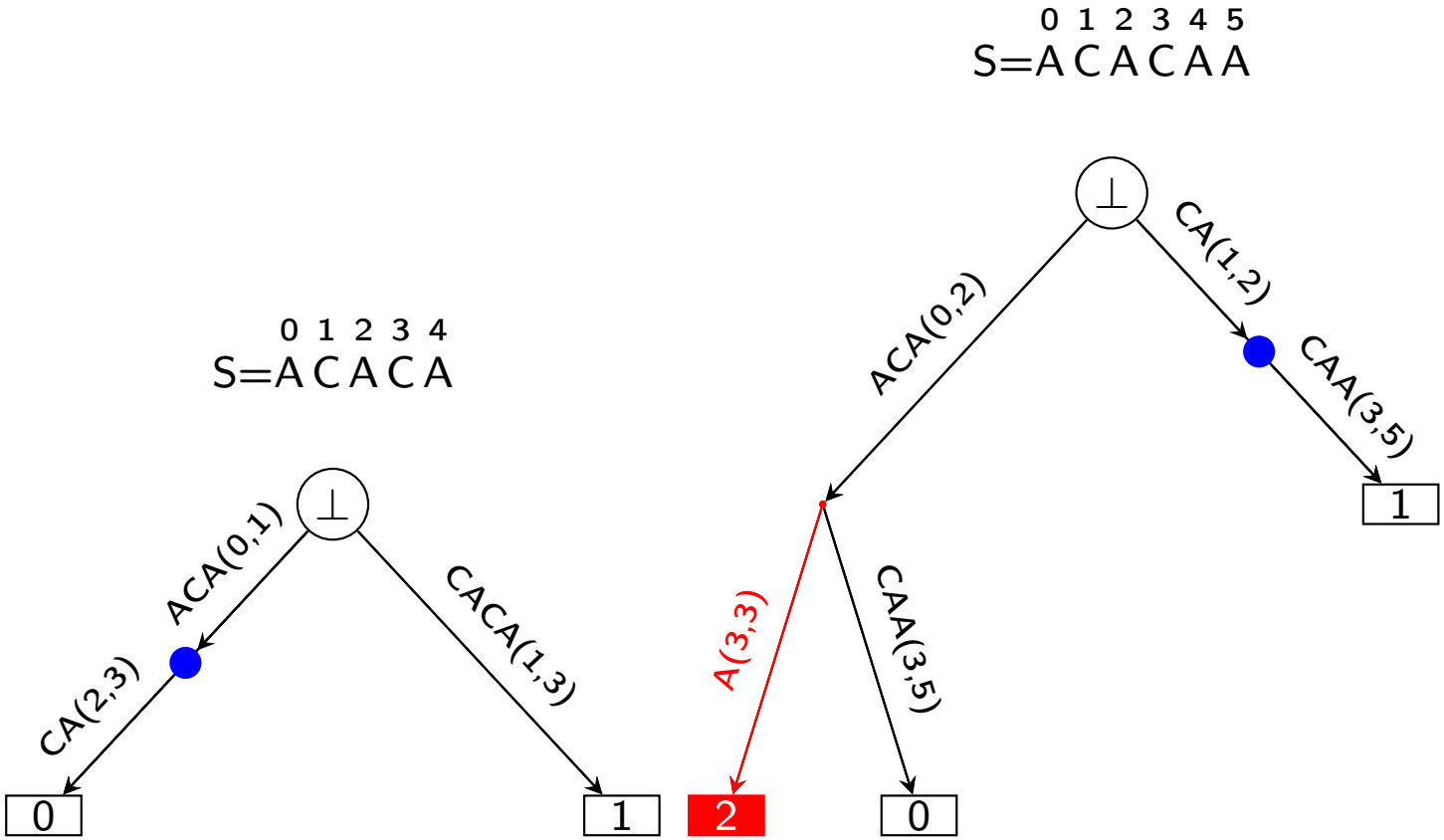


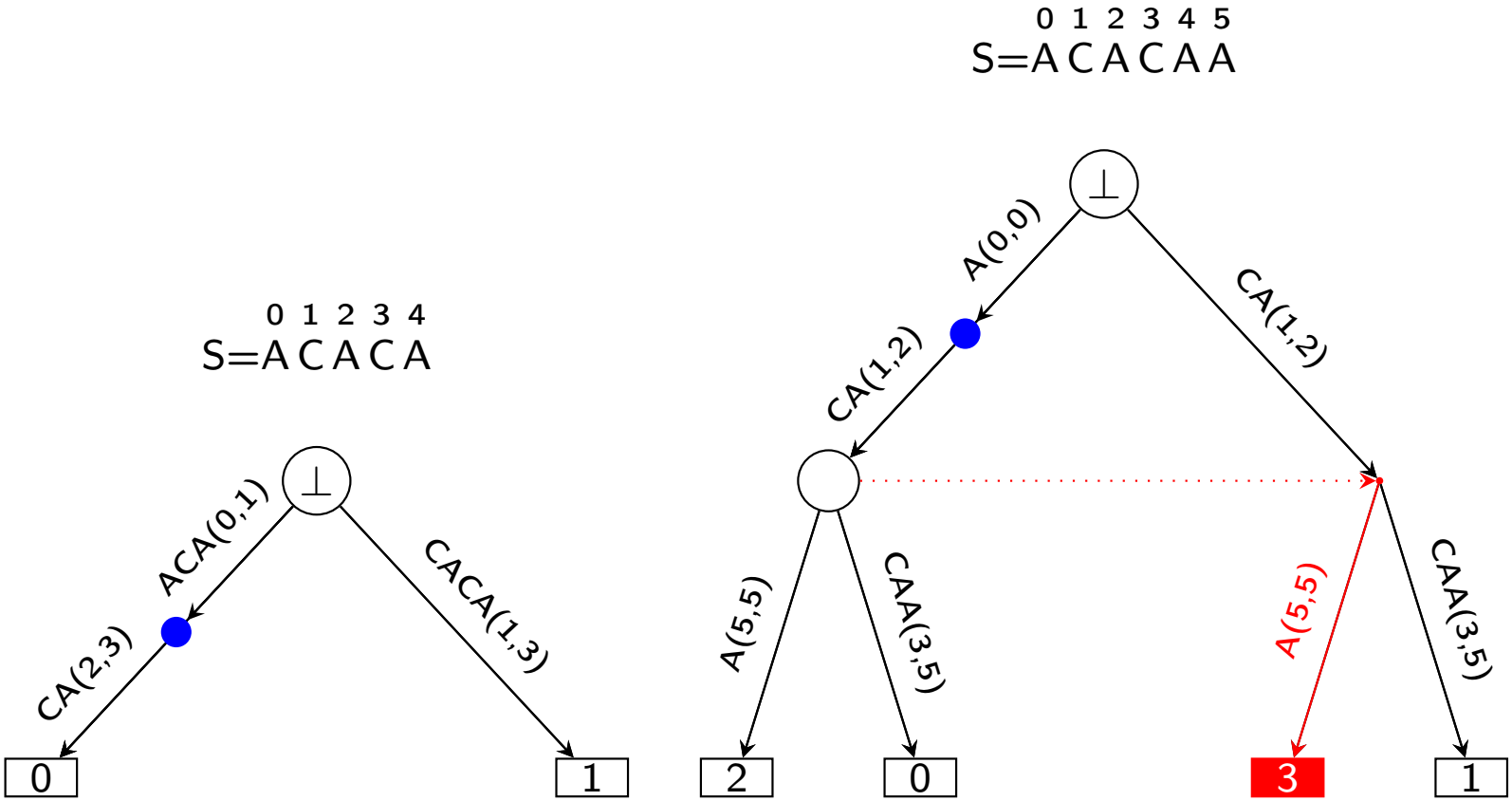
Ukkonen construction algorithm of the suffix tree

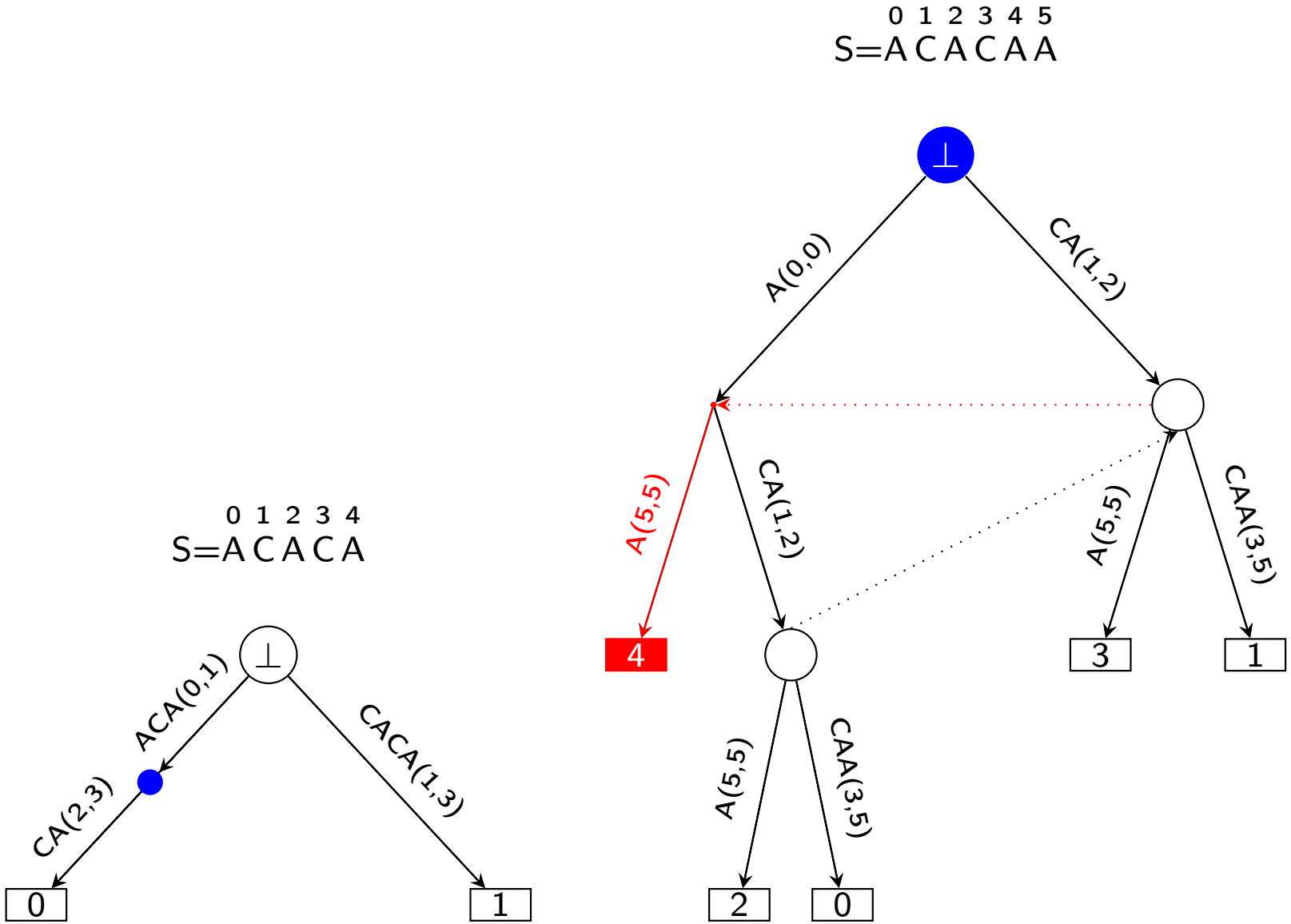


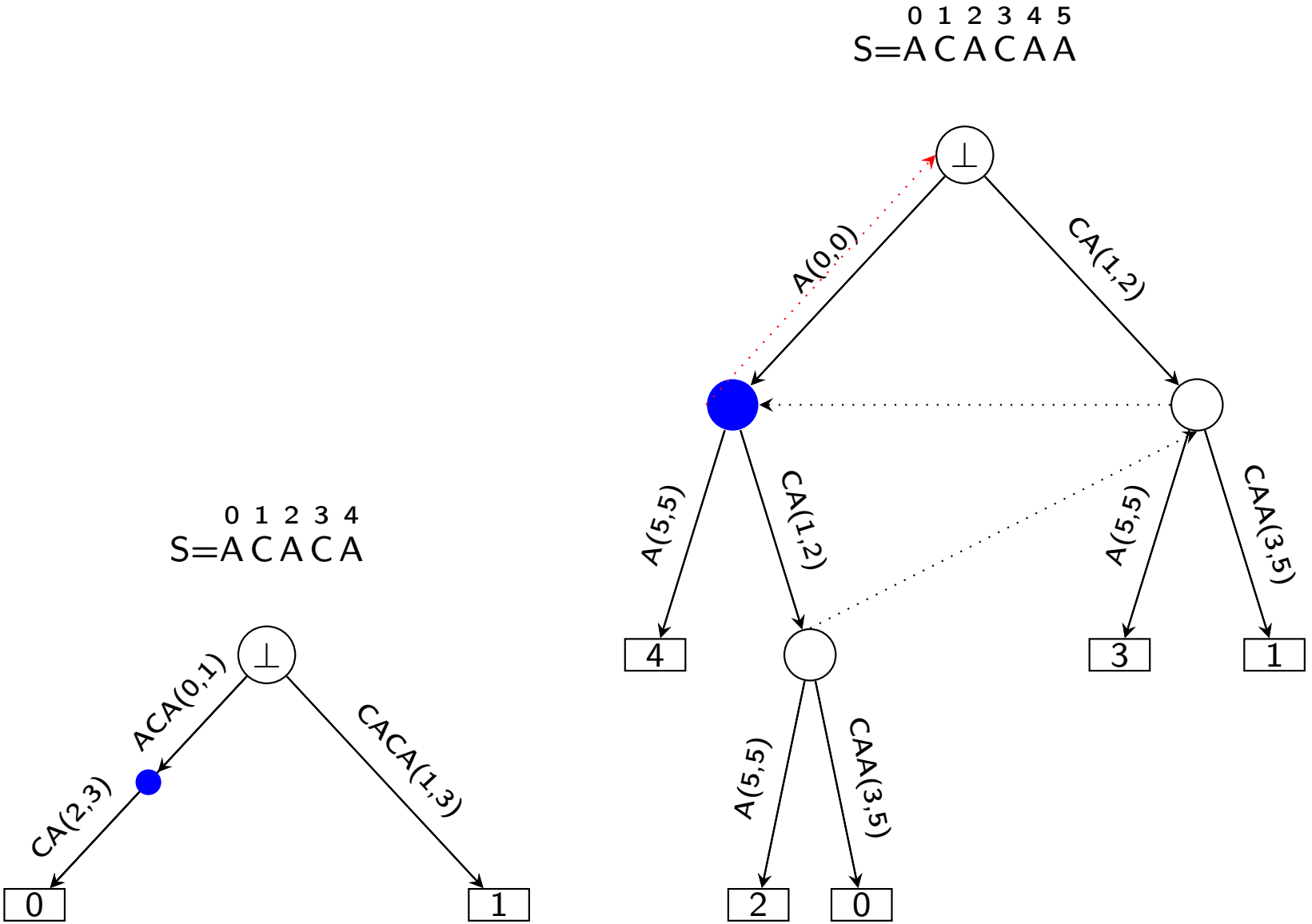

0 1 2
S=A C A
⊥
A(0,0)
CA(1,2)
CA(1,2)
0
1
0 1 2 3
S=A C A C
⊥
AC(0,1)
CAC(1,3)
AC(2,3)
0
1

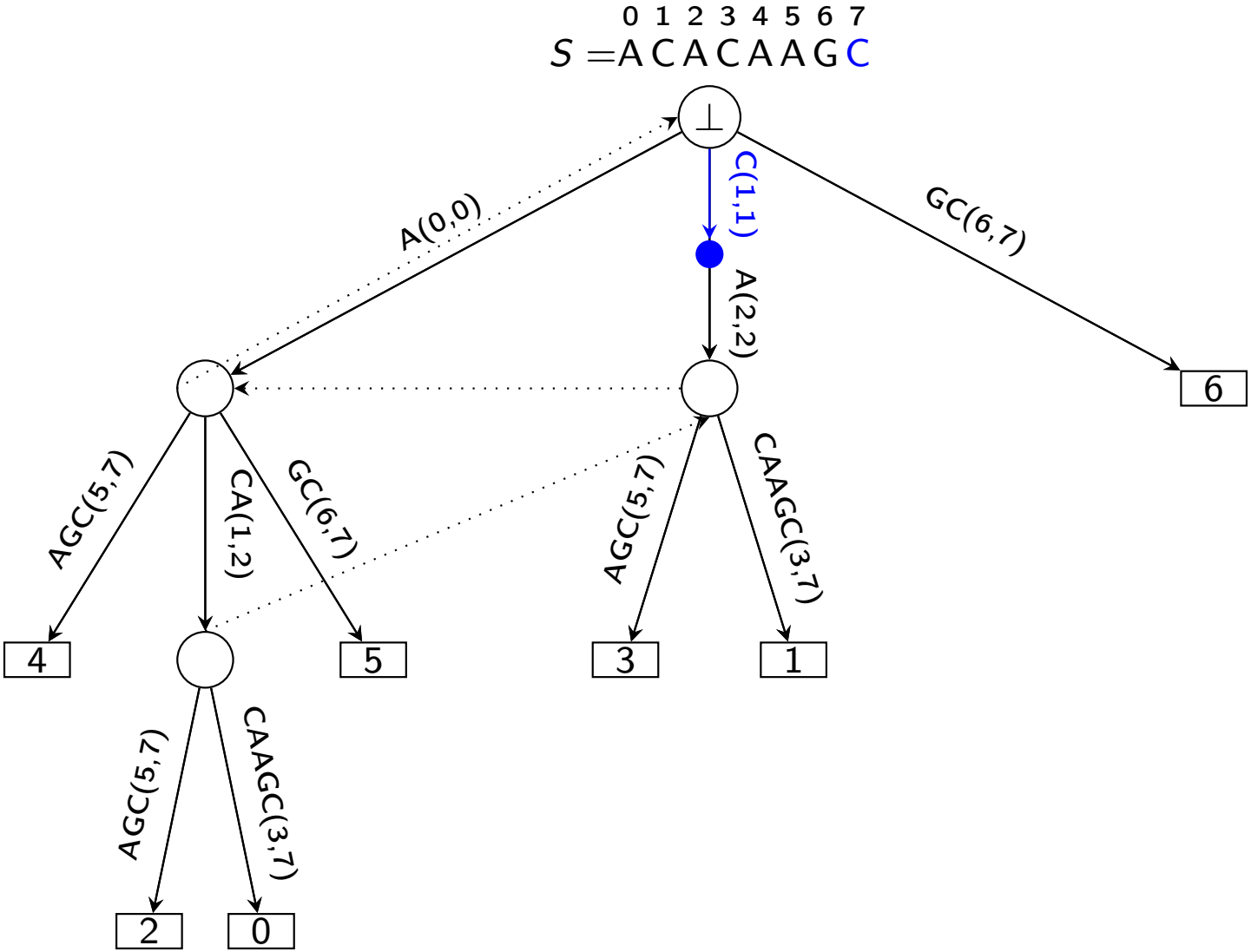

0 1 2
S=A C A
⊥
A(0,0)
CA(1,2)
CA(1,2)
0
1
0 1 2 3
S=A C A C
⊥
AC(0,1)
CAC(1,3)
AC(2,3)
0
1
0 1 2 3 4
S=A C A C A
⊥
ACA(0,1)
CACA(1,3)
CA(2,3)
0
1


0 1 2 3 4
S=A C A C A
⊥
ACA(0,1)
CACA(1,3)
CA(2,3)
0
1
0 1 2 3 4 5
S=A C A C A A
⊥
ACA(0,2)
CA(1,2)
CAA(3,5)
1
A(3,3)
CAA(3,5)
2
0


0 1 2 3 4
S=A C A C A
⊥
ACA(0,1)
CACA(1,3)
CA(2,3)
0
1
0 1 2 3 4 5
S=A C A C A A
⊥
A(0,0)
CA(1,2)
CA(1,2)
A(5,5)
CAA(3,5)
A(5,5)
CAA(3,5)
2
0
3
1


0 1 2 3 4 5
S=A C A C A A
⊥
A(0,0)
CA(1,2)
A(5,5)
CA(1,2)
A(5,5)
CAA(3,5)
4
3
1
A(5,5)
CAA(3,5)
2
0
0 1 2 3 4
S=A C A C A
⊥
ACA(0,1)
CACA(1,3)
CA(2,3)
0
1

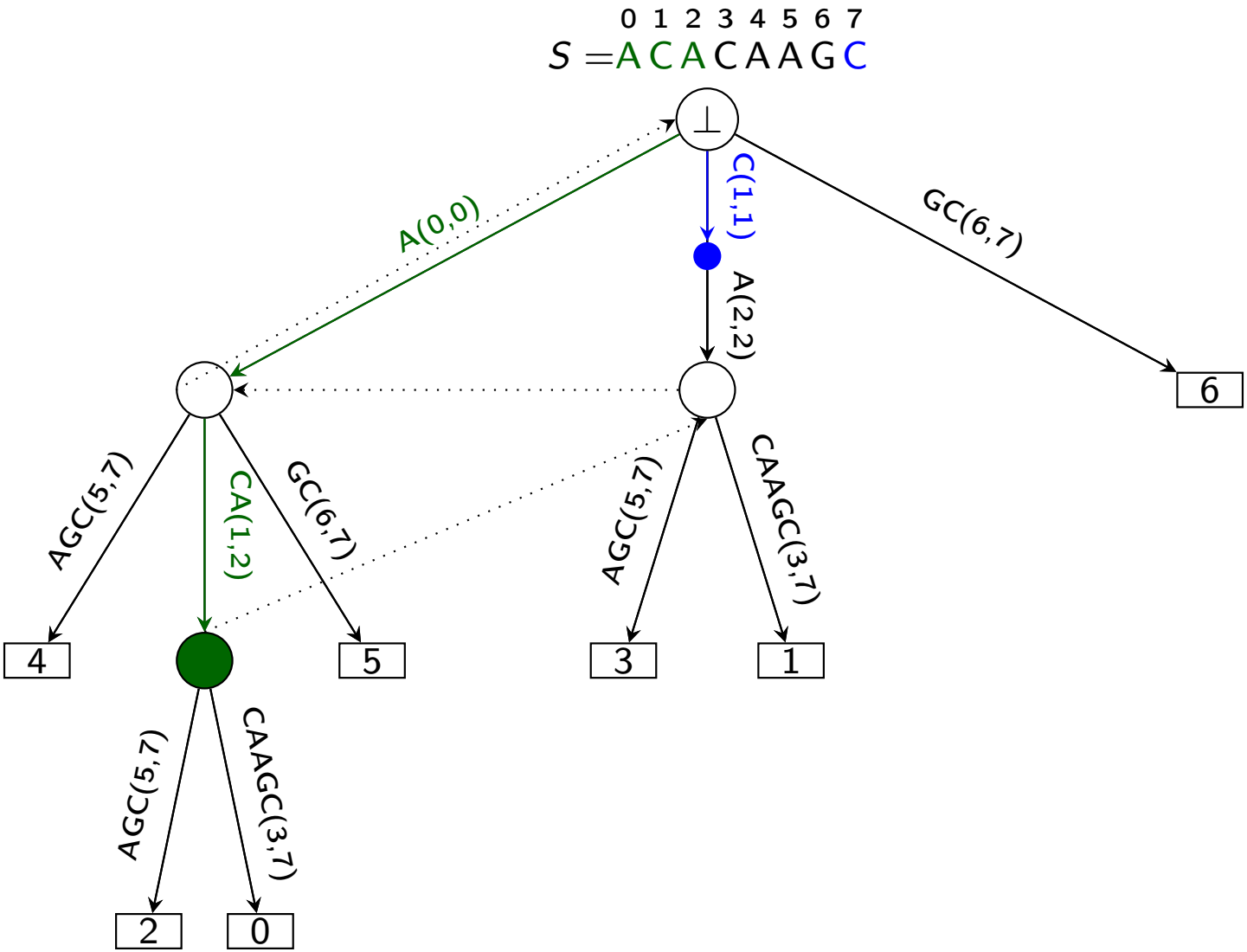

0 1 2 3 4 5
S=A C A C A A
A(0,0)
CA(1,2)
A(5,5)
CA(1,2)
A(5,5)
CAA(3,5)
4
A(5,5)
CAA(3,5)
3
1
2
0
0 1 2 3 4
S=A C A C A
ACA(0,1)
CACA(1,3)
CA(2,3)
0
1


0 1 2 3 4 5 6 7
S = A C A C A A G C
⊥
A(0,0)
C(1,1)
A(2,2)
GC(6,7)
6
AGC(5,7)
CA(1,2)
GC(6,7)
4
5
AGC(5,7)
CAAGC(3,7)
3
1
AGC(5,7)
CAAGC(3,7)
2
0

The suffix tree for a sliding window

The suffix tree for a sliding window, Senft 2005

- Remove the leftmost letter,
- Update edge labels.


0 1 2 3 4 5 6 7
S = A C A C A A G C
⊥
A(0,0)
C(1,1)
GC(6,7)
A(2,2)
6
AGC(5,7)
CA(1,2)
GC(6,7)
AGC(5,7)
CAAGC(3,7)
4
5
3
1
AGC(5,7)
CAAGC(3,7)
2
0

Minimal absent words over a sliding window

We have adapted Senft algorithm to compute minimal absent words.

Minimal absent words over a sliding window

We have adapted Senft algorithm to compute minimal absent words.

- Add on the tree the information of the BWT (the set of letters that precede each factor),

Minimal absent words over a sliding window

We have adapted Senft algorithm to compute minimal absent words.

- Add on the tree the information of the BWT (the set of letters that precede each factor),
- Add the set of minimal absent words

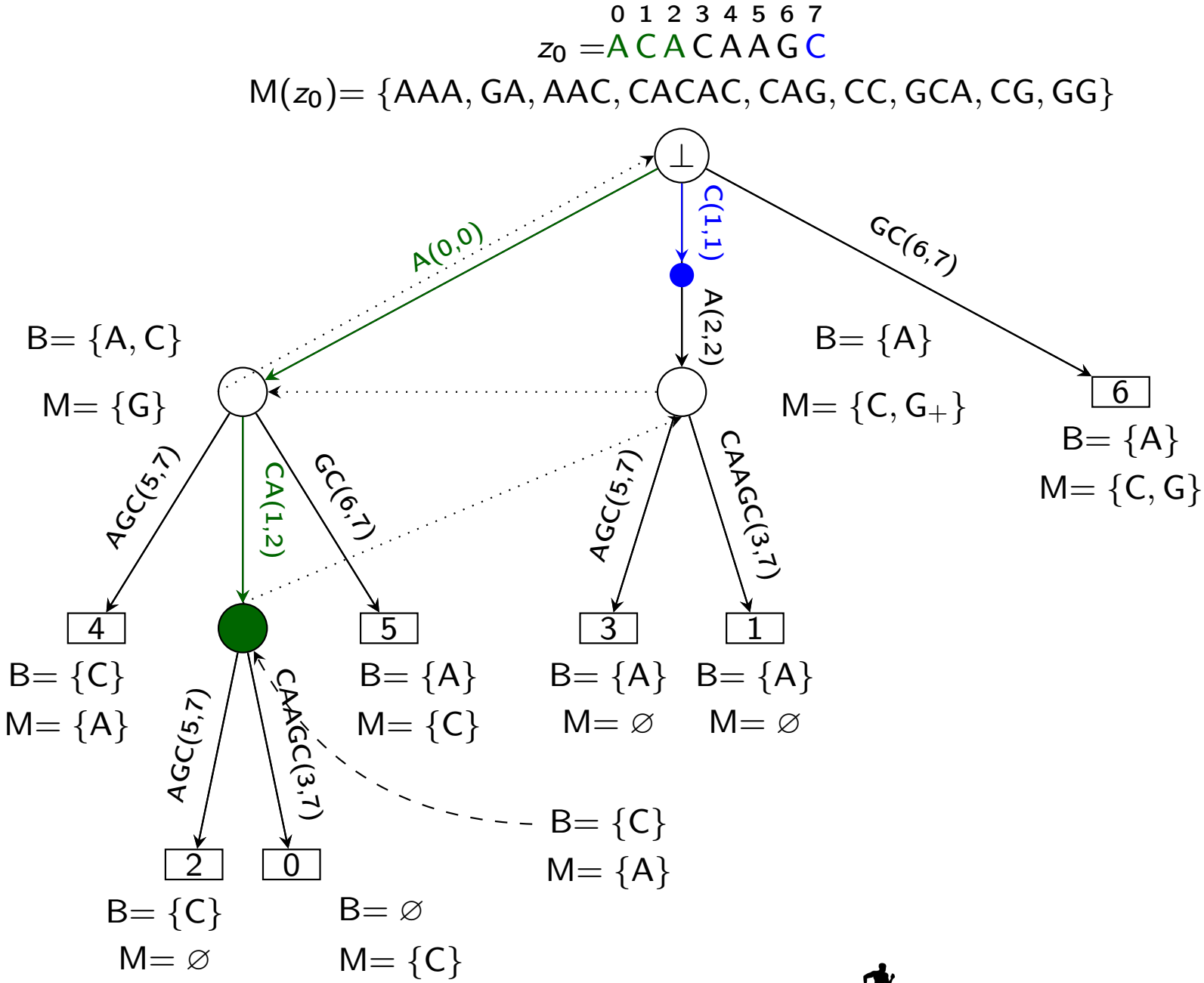
Minimal absent words over a sliding window

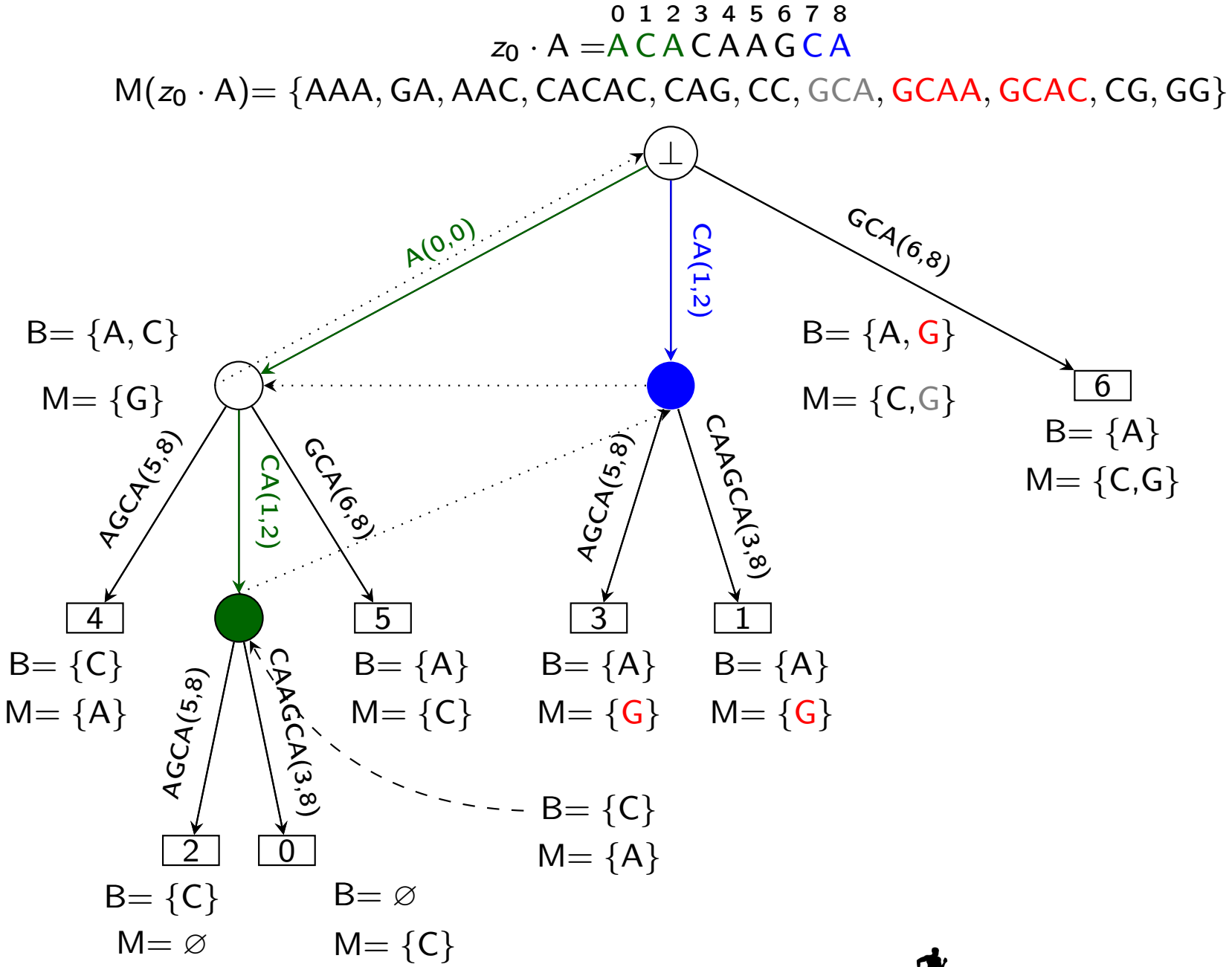
We have adapted Senft algorithm to compute minimal absent words.

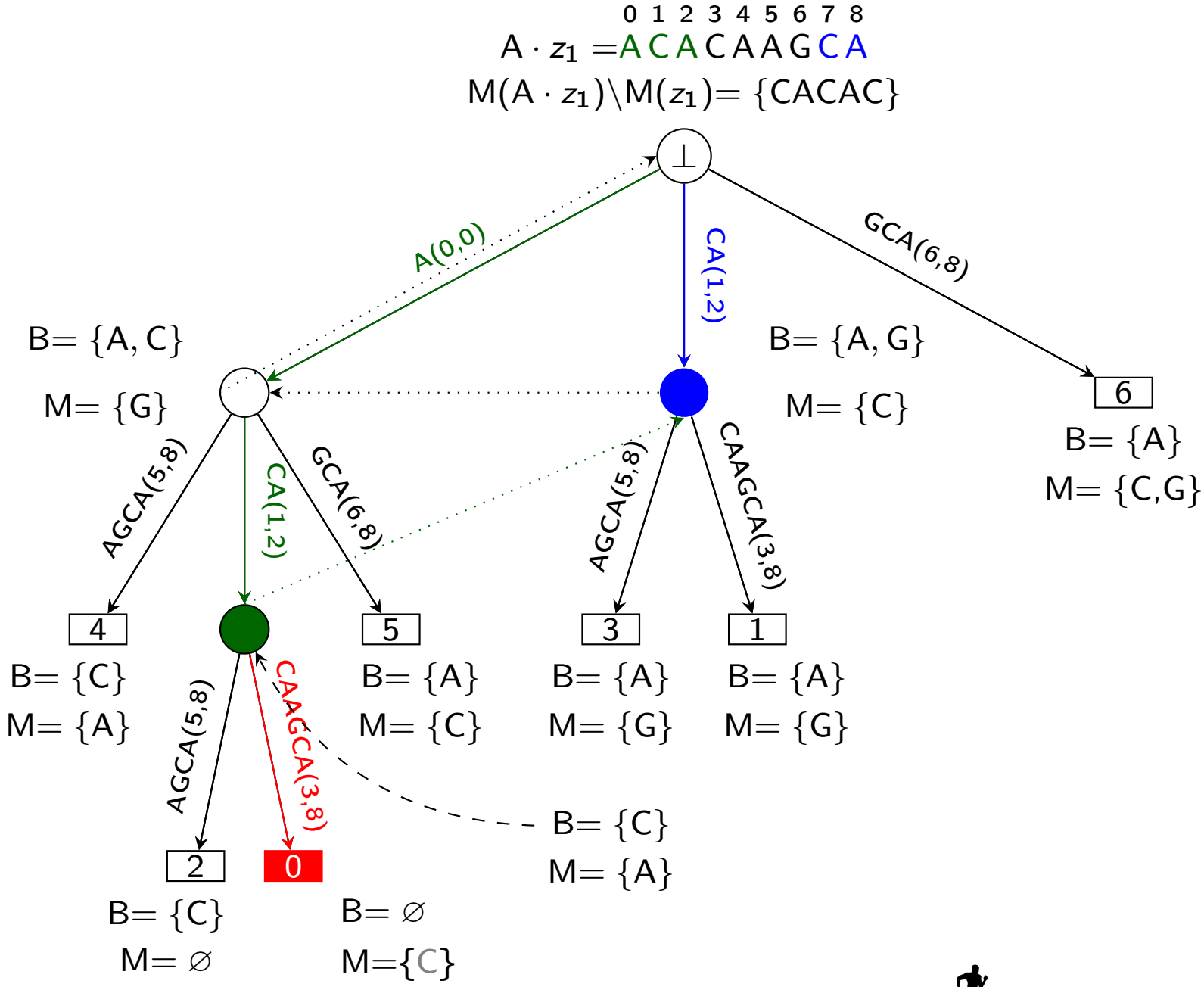
- Add on the tree the information of the BWT (the set of letters that precede each factor),
- Add the set of minimal absent words

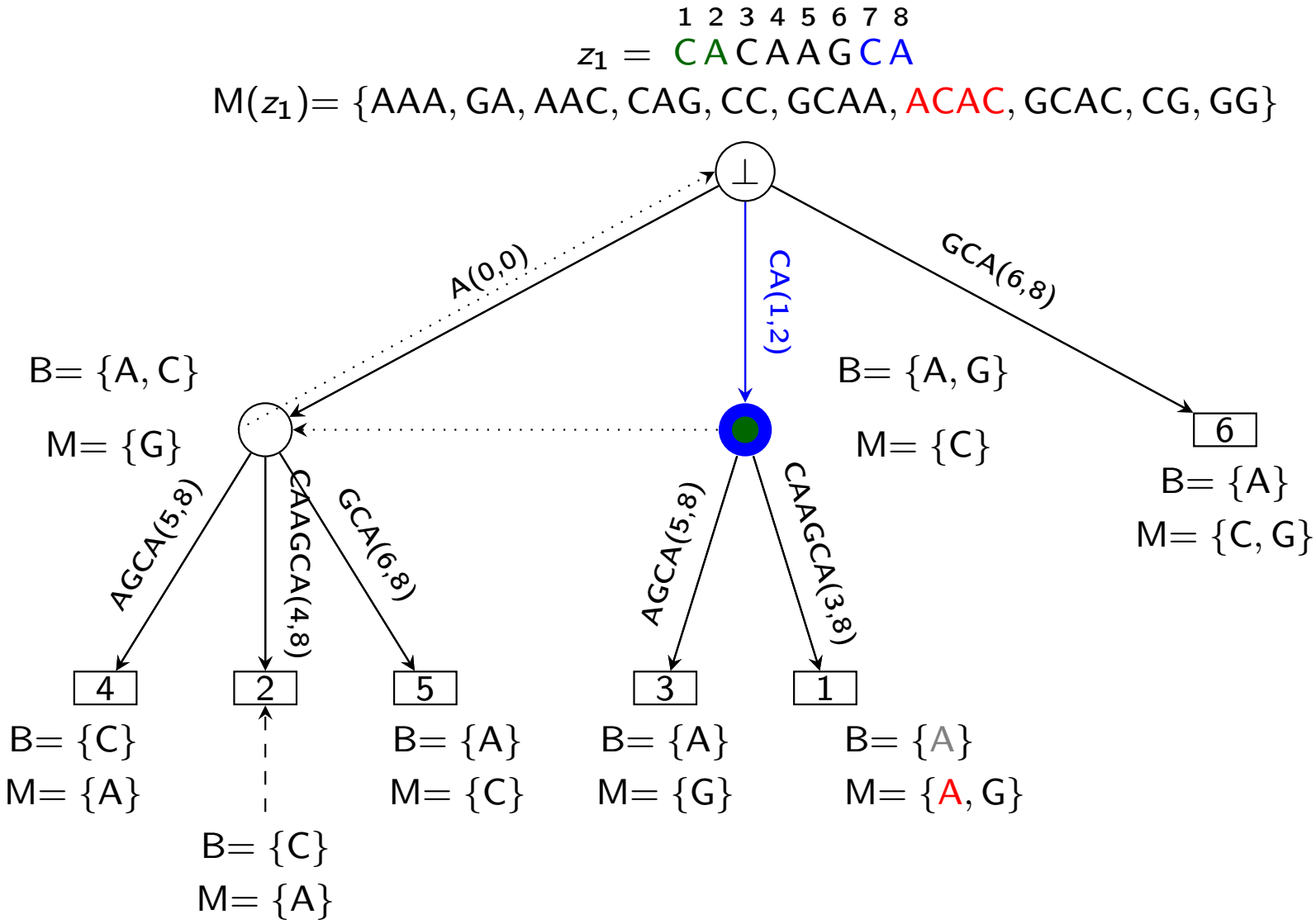
The mapping f is an injection

$f : M(z) \to \Sigma(z) \times V(z)$ define by $f(aub) = (a, v_{ub})$,
where $a \in \Sigma$ and v_{ub} is the node corresponding to ub.


0 1 2 3 4 5 6 7
z_0 = A C A C A A G C
M(z_0)= {AAA, GA, AAC, CACAC, CAG, CC, GCA, CG, GG}
⊥
A(0,0)
C(1,1)
A(2,2)
GC(6,7)
B= {A, C}
M= {G}
B= {A}
M= {C, G+}
6
B= {A}
M= {C, G}
AGC(5,7)
CA(1,2)
GC(6,7)
AGC(5,7)
CAAGC(3,7)
4
5
3
1
B= {C}
M= {A}
B= {A}
M= {C}
B= {A}
M= ∅
B= {A}
M= ∅
AGC(5,7)
CAAGC(3,7)
2
0
B= {C}
M= {A}
B= {C}
M= ∅
B= ∅
M= {C}


0 1 2 3 4 5 6 7 8
$z_0 \cdot A$ = A C A C A A G C A
M($z_0 \cdot A$)= {AAA, GA, AAC, CACAC, CAG, CC, GCA, GCAA, GCAC, CG, GG}
⊥
A(0,0)
CA(1,2)
GCA(6,8)
B= {A, C}
M= {G}
B= {A, G}
M= {C,G}
6
B= {A}
M= {C,G}
AGCA(5,8)
CA(1,2)
GCA(6,8)
AGCA(5,8)
CAAGCA(3,8)
4
5
3
1
B= {C}
M= {A}
B= {A}
M= {C}
B= {A}
M= {G}
B= {A}
M= {G}
AGCA(5,8)
CAAGCA(3,8)
2
0
B= {C}
M= {A}
B= {C}
M= ∅
B= ∅
M= {C}


0 1 2 3 4 5 6 7 8
A · z1 =A C A C A A G C A
M(A · z1)\M(z1)= {CACAC}
⊥
A(0,0)
CA(1,2)
GCA(6,8)
B= {A, C}
M= {G}
B= {A, G}
M= {C}
6
B= {A}
M= {C,G}
AGCA(5,8)
CA(1,2)
GCA(6,8)
AGCA(5,8)
CAAGCA(3,8)
4
5
3
1
B= {C}
M= {A}
B= {A}
M= {C}
B= {A}
M= {G}
B= {A}
M= {G}
AGCA(5,8)
CAAGCA(3,8)
2
0
B= {C}
M= {A}
B= {C}
M= ∅
B= ∅
M={C}


1 2 3 4 5 6 7 8
z_1 = C A C A A G C A
M(z_1)= {AAA, GA, AAC, CAG, CC, GCAA, ACAC, GCAC, CG, GG}
⊥
A(0,0)
CA(1,2)
GCA(6,8)
B= {A, C}
M= {G}
B= {A, G}
M= {C}
6
B= {A}
M= {C, G}
AGCA(5,8)
CAAGCA(4,8)
GCA(6,8)
AGCA(5,8)
CAAGCA(3,8)
4
2
5
3
1
B= {C}
M= {A}
B= {C}
M= {A}
B= {A}
M= {C}
B= {A}
M= {G}
B= {A}
M= {A, G}

Applications to on-line pattern matching

Minimal absent words over a sliding window

For a sequence S of size n and a window of size m we compute :
$\forall i, 0 \leq i \leq n - m, M(S[i \,..\, i + m - 1])$,
in time $\mathcal{O}(n)$ and in space $\mathcal{O}(m)$.

Applications to on-line pattern matching

Minimal absent words over a sliding window

For a sequence S of size n and a window of size m we compute :
$\forall i, 0 \leq i \leq n - m, M(S[i \,..\, i + m - 1])$,
in time $\mathcal{O}(n)$ and in space $\mathcal{O}(m)$.

Length Weighted Index (LWI), introduced by Chairungsee in 2012

Metric based on the symmetric difference of minimal absent words sets

$$\mathrm{LWI}(M(x), M(y)) = \sum_{w \in M(x) \triangle M(y)} \frac{1}{|w|^2}.$$

Applications to on-line pattern matching

Minimal absent words over a sliding window

For a sequence S of size n and a window of size m we compute :
$\forall i, 0 \leq i \leq n-m, M(S[i\,..\,i+m-1])$,
in time $\mathcal{O}(n)$ and in space $\mathcal{O}(m)$.

Length Weighted Index (LWI), introduced by Chairungsee in 2012

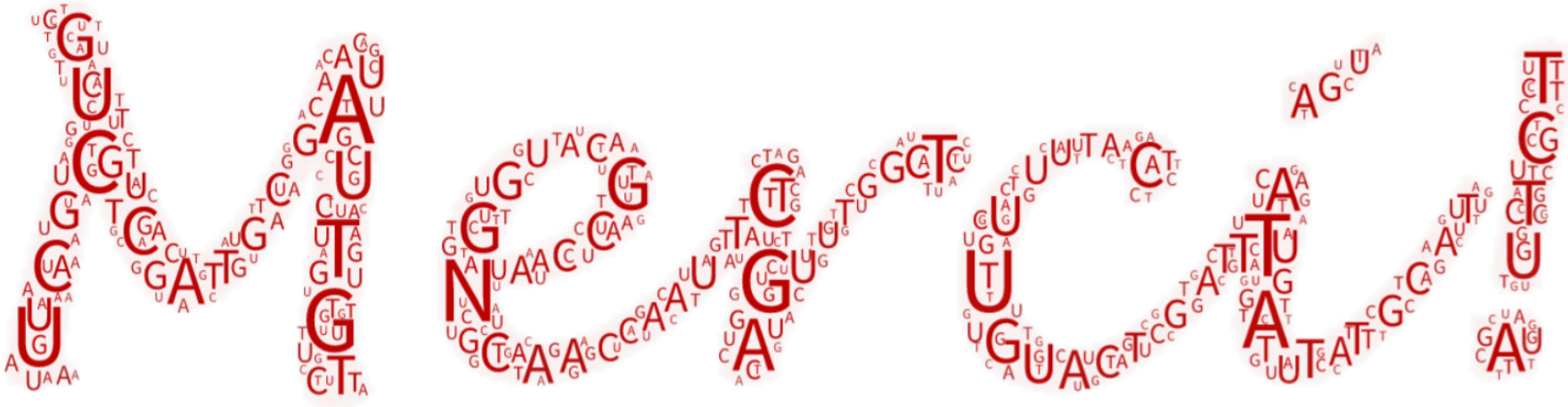
Metric based on the symmetric difference of minimal absent words sets

$$\mathrm{LWI}(M(x), M(y)) = \sum_{w \in M(x) \triangle M(y)} \frac{1}{|w|^2}.$$

$\rightarrow$ We obtain the position of minimal distance.

Futur works

- Implement the algorithm over a sliding window,
- Compare the results in Bioinformatics for reads alignment.


MERCI

LOB

- Hubert Becker
- Hannu Myllykallio
- Roxane Lestini
- Yoann Collien
- et tous les autres

LIX

- Mireille Régnier
- Yann Ponty
- Philippe Chassignet
- Amélie Héliou
- Afaf Saaidi
- Juraj Michalik
- et tous les autres

Université de Rouen

- Laurent Mouchard

King's College London

- Solon Pissis
- Carl Barton

Université d'Helsinki

- Simon Puglisi

Université Paris Est

- Maxime Crochemore
- Gregory Kucherov

ENS Paris

- Yann Ramusat

Algorithms to compute minimal absent words

References	Structures	Drawbacks
Crochemore et al., 1998	Suffix Automata	Expensive in space
Belazzougui et al. 2013	Bidirectionnal BWT	No implementation available
Ota et al. 2014	Suffix tree, dynamic approach	Quadratic in time
Barton et al. 2014	Suffix Array	Linear, fastest available
Belazzougui et al. 2015	BWT & complementary structures	No implementation