String interleaving

ALGORITHMICS

Vicente García Díaz
garciavicente@uniovi.es

What is the idea?

- He have 3 strings of characters
  - A with |A| = n
  - B with |B| = m
  - C with |C| = n + m

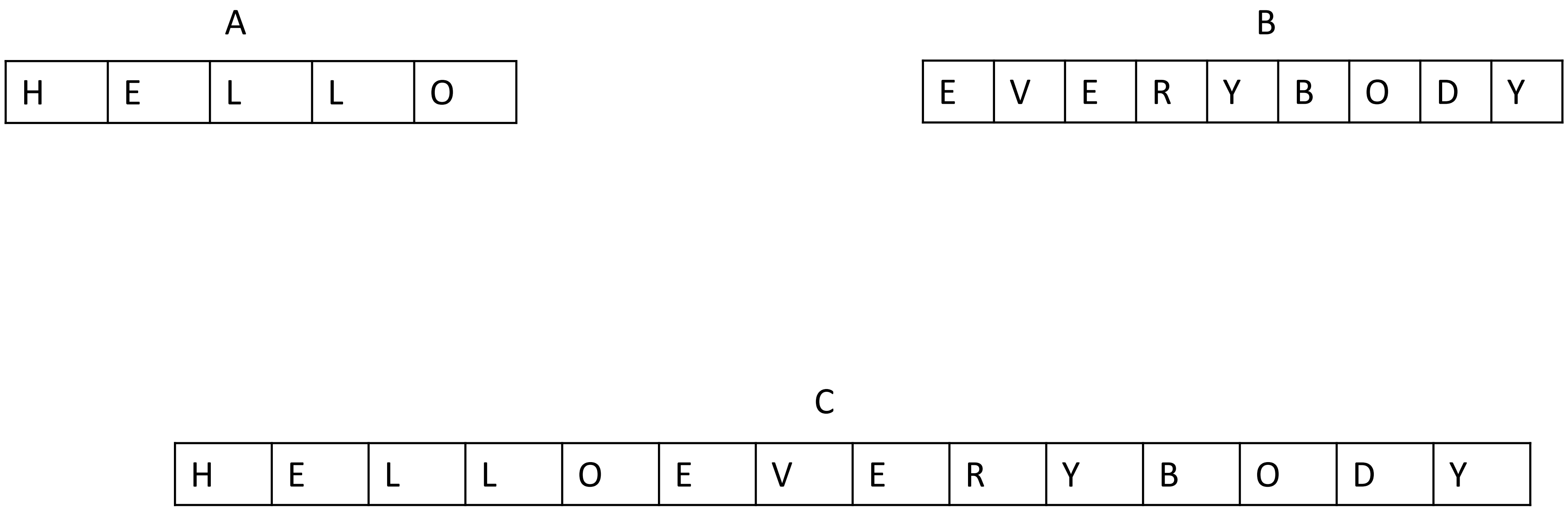
- **C** is said to be a shuffle of **A** and **B** iff **C** can be created by interleaving the characters from **A** and **B** in a way that maintains the left-to-right ordering of the characters from each string

Greedy Algorithm

- Propose a Greedy Algorithm with a linear complexity to conclude that **C** is a shuffle of **A** and **B**
  - A = HELLO
  - B = EVERYBODY
  - C = HELLOEVERYBODY

```
FOR EACH CHARACTER OF C
    WE TAKE A CHARACTER OF A IF POSSIBLE
    IF NOT, WE TAKE A CHARACTER OF B IF POSSIBLE
    IF NOT, RETURN FALSE
RETURN TRUE
```

Development



Greedy Algorithm (II)

- Use your previos Greedy Algorithm to conclude that **C** is a shuffle of **A** and **B**
  - A = HELLO
  - B = EVERYBODY
  - C = HEEVERYBLLOODY

Development

A

H	E	L	L	O

B

E	V	E	R	Y	B	O	D	Y

C

H	E	E	V	E	R	Y	B	L	L	O	O	D	Y

Greedy Algorithm (III)

- Use your previos Greedy Algorithm to conclude that **C** is a shuffle of **A** and **B**
  - A = HELLO
  - B = EVERYBODY
  - C = HEVEERYLBLOODY

Development

A

H	E	L	L	O

B

E	V	E	R	Y	B	O	D	Y

C

H	E	V	E	E	R	Y	L	B	L	O	O	D	Y

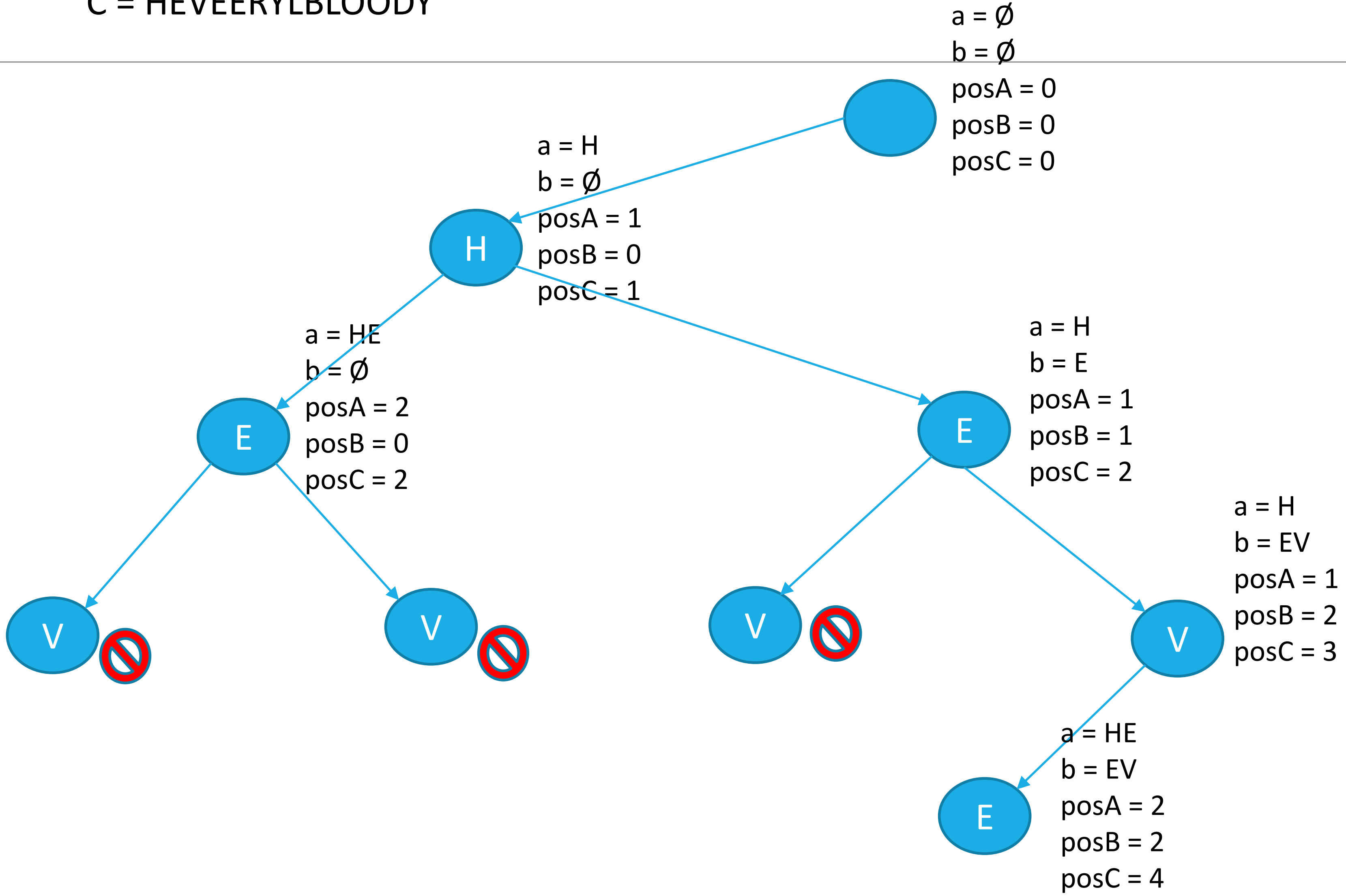
Divide and Conquer

- Propose a Divide and Conquer Algorithm to conclude that **C** is a shuffle of **A** and **B**
  - A = HELLO
  - B = EVERYBODY
  - C = HEVEERYLBLOODY

- We have two possibilities
  - If the first character of **C** matches the first character of **A**, we move one character ahead in **A** and **C** and recursively check
  - If the first character of **C** matches the first character of **B**, we move one character ahead in **B** and **C** and recursively check
  - If any of the above cases is **true**, we return **true**, **false** otherwise

Development

A = HELLO
B = EVERYBODY
C = HEVEERYLBLOODY

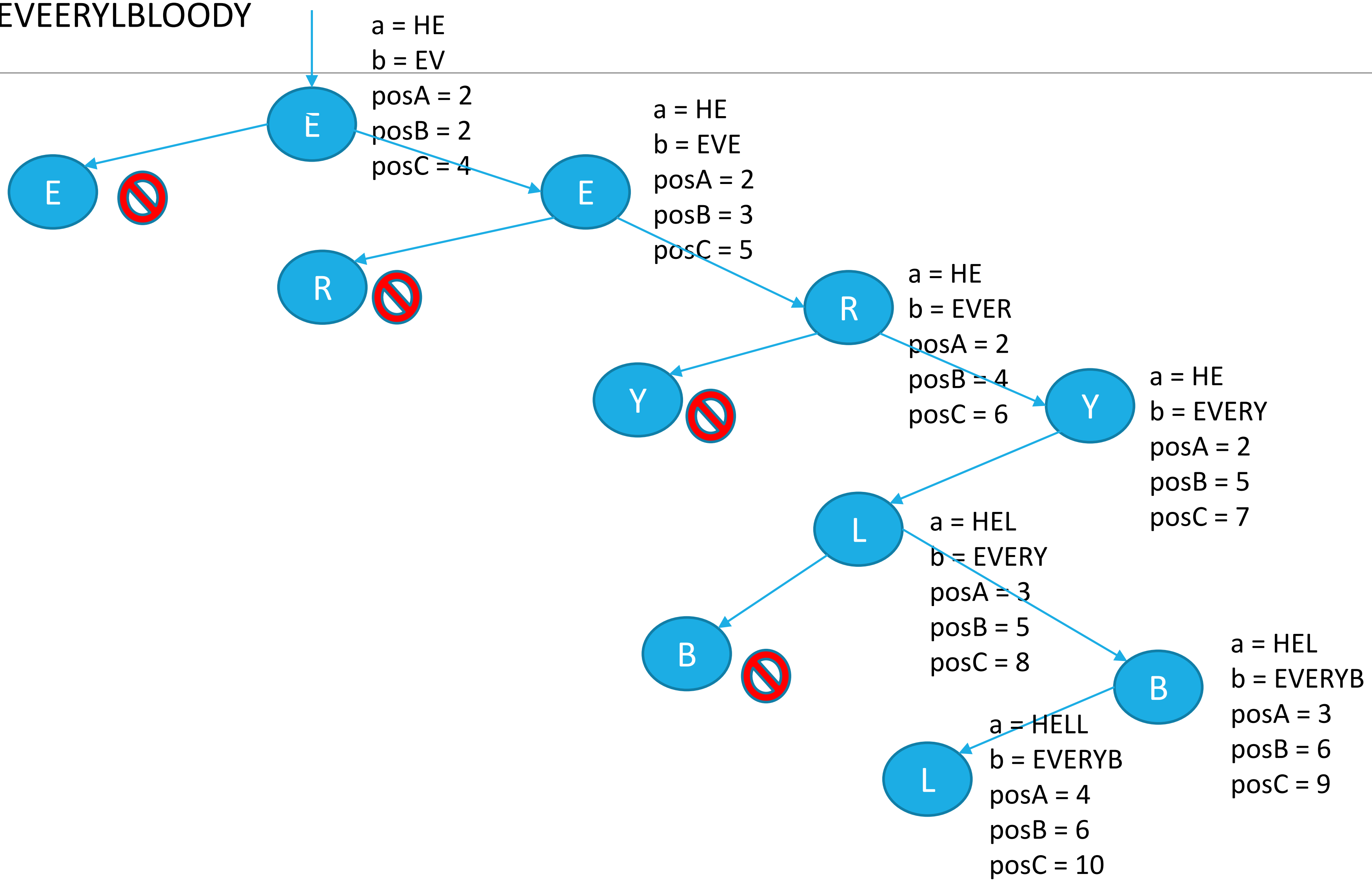


Development (II)

A = HELLO

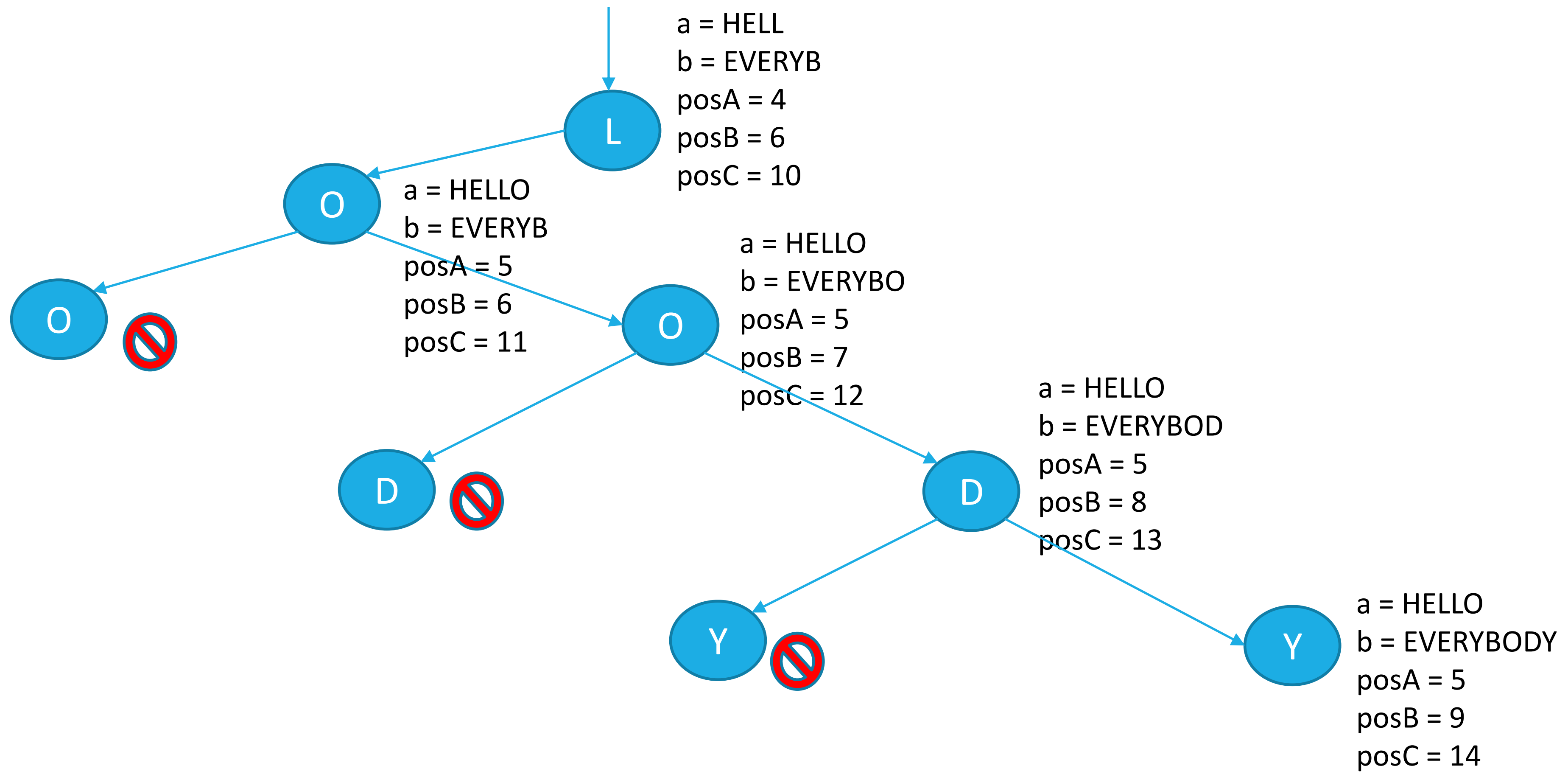
B = EVERYBODY

C = HEVEERYLBLOODY



A = HELLO
B = EVERYBODY
C = HEVEERYLBLOODY

Development (III)



Dynamic Programming

- Propose a Dynamic Programming Algorithm to conclude that **C** is a shuffle of **A** and **B**
  - A = HELLO
  - B = EVERYBODY
  - C = EVERYHELBODYLO

	Ø	E	V	E	R	Y	B	O	D	Y
Ø										
H										
E										
L										
L										
O										

Dynamic Programming (II)

- Propose a Dynamic Programming Algorithm to conclude that **C** is a shuffle of **A** and **B**
  - A = HELLO
  - B = EVERYBODY
  - C = EVERYHELBODYLO

j

i

	Ø	E	V	E	R	Y	B	O	D	Y
Ø										
H										
E										
L										
L										
O										

$$T[i, j] \begin{cases} \text{If } i = j = \varnothing & \text{TRUE} \\ \text{If } A_i \neq C_{i+j} \text{ AND } B_j \neq C_{i+j} & \text{FALSE} \\ \text{If } A_i = C_{i+j} \text{ AND } B_j \neq C_{i+j} & T[i-1, j] \\ \text{If } A_i \neq C_{i+j} \text{ AND } B_j = C_{i+j} & T[i, j-1] \\ \text{If } A_i = B_j = C_{i+j} & T[i-1, j] \text{ OR } T[i, j-1] \end{cases}$$

Development

- A = HELLO
- B = EVERYBODY
- C = EVERYHELBODYLO

$$T[i, j] = \begin{cases} \text{TRUE} & \text{If } i = j = \emptyset \\ \text{FALSE} & \text{If } A_i \neq C_{i+j} \text{ AND } B_j \neq C_{i+j} \\ T[i-1, j] & \text{If } A_i = C_{i+j} \text{ AND } B_j \neq C_{i+j} \\ T[i, j-1] & \text{If } A_i \neq C_{i+j} \text{ AND } B_j = C_{i+j} \\ T[i-1, j] \text{ OR } T[i, j-1] & \text{If } A_i = B_j = C_{i+j} \end{cases}$$

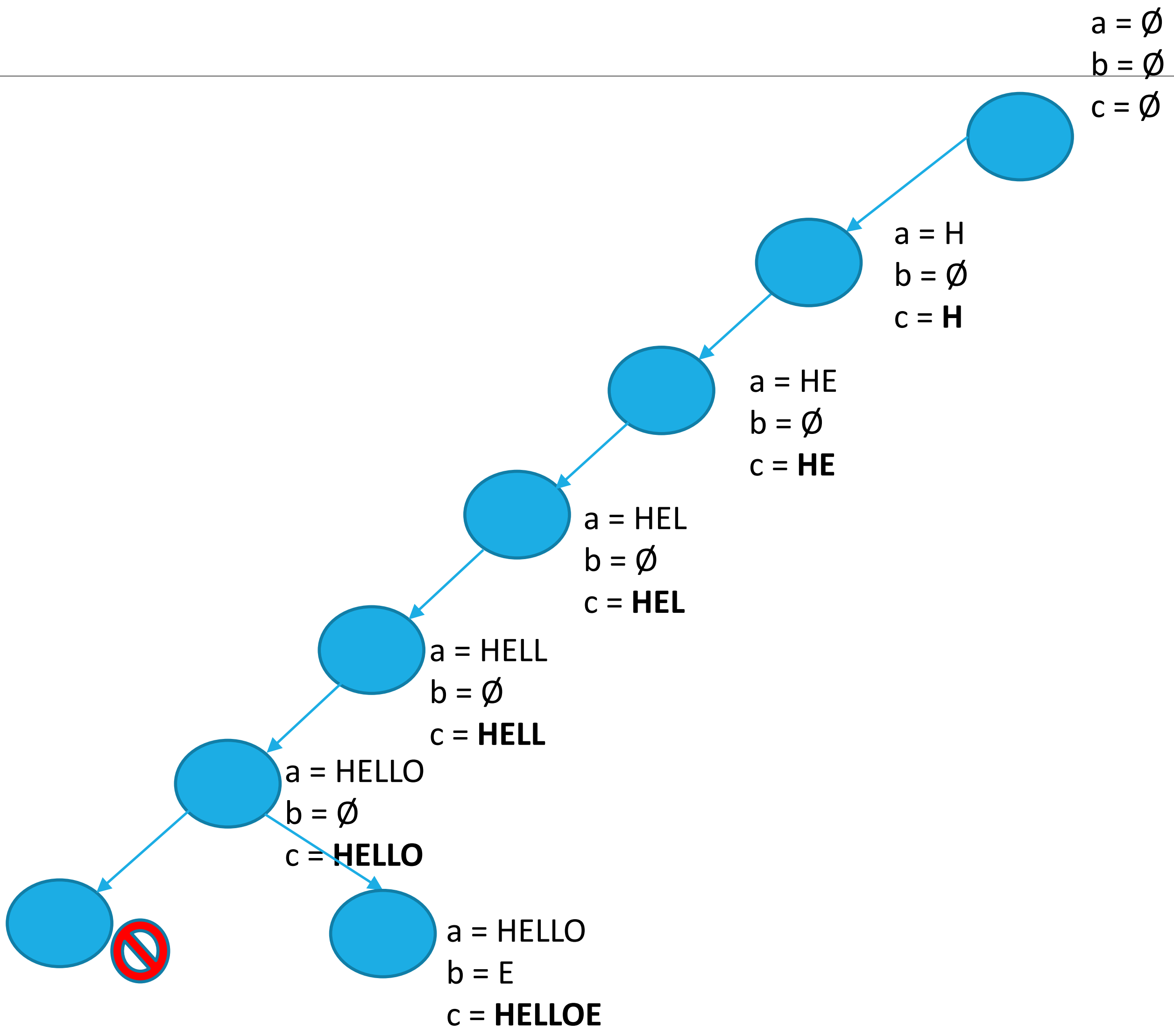
	Ø	E	V	E	R	Y	B	O	D	Y
Ø										
H										
E										
L										
L										
O										

All possible shuffles C from A and B

- Propose an Algorithm to generate all posible shuffles **C** from **A** and **B**
  - A = HELLO
  - B = EVERYBODY

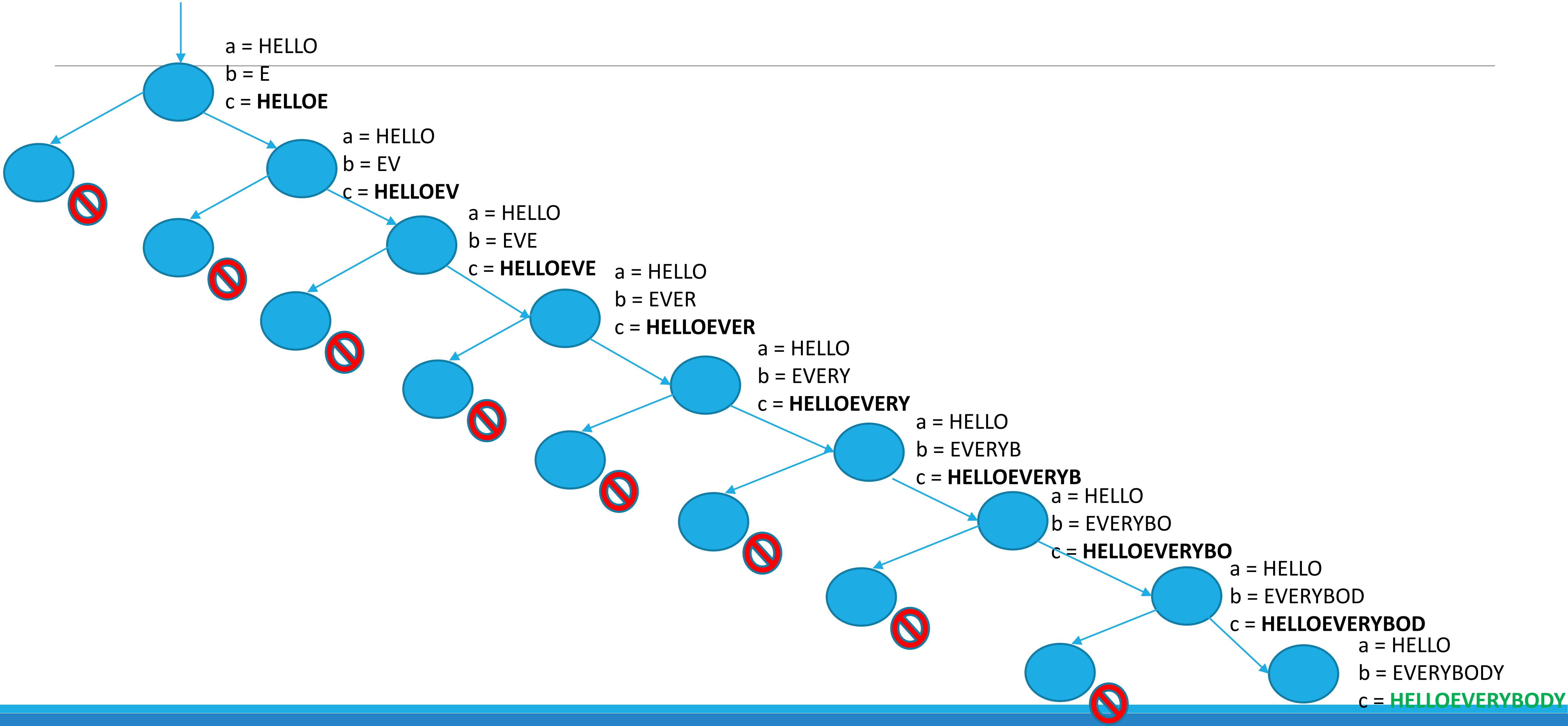
Development

A = HELLO
B = EVERYBODY



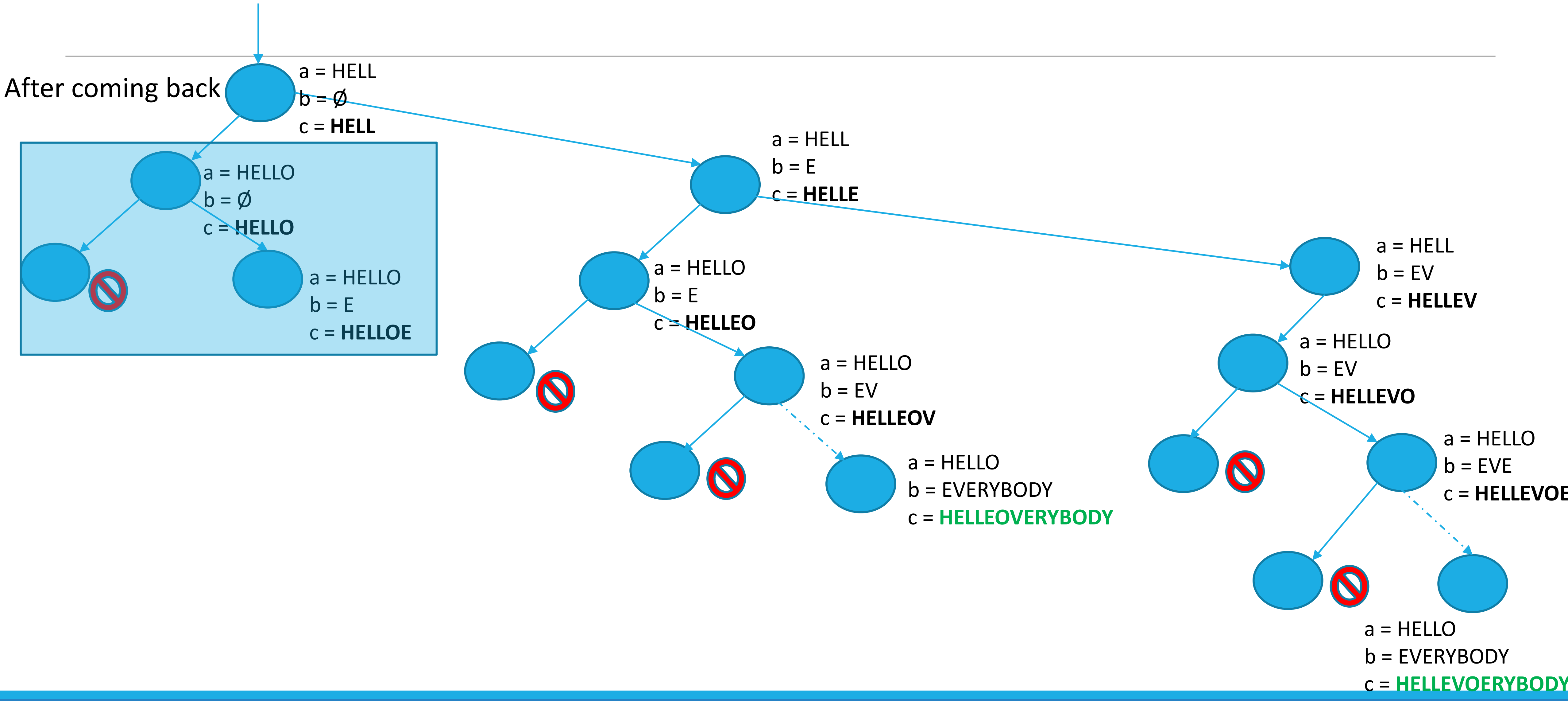
Development (II)

A = HELLO
B = EVERYBODY



Development (III)

A = HELLO
B = EVERYBODY


After coming back
a = HELL
b = Ø
c = HELL
a = HELLO
b = Ø
c = HELLO
a = HELLO
b = E
c = HELLOE
a = HELL
b = E
c = HELLE
a = HELLO
b = E
c = HELLEO
a = HELLO
b = EV
c = HELLEOV
a = HELLO
b = EVERYBODY
c = HELLEOVERYBODY
a = HELL
b = EV
c = HELLEV
a = HELLO
b = EV
c = HELLEVO
a = HELLO
b = EVE
c = HELLEVOE
a = HELLO
b = EVERYBODY
c = HELLEVOERYBODY