Tries

Shusen Wang

Motivations

Motivations

- We have a dictionary, e.g., {“a”, “absolute”, “accelerate”, …, “zoo”}.

Task 1: Search

- Given a query word, e.g., query = “algorithm”.
- Is the query word in the dictionary?

Motivations

- We have a dictionary, e.g., {“a”, “absolute”, “accelerate”, …, “zoo”}.

Task 1: Search

Task 2: Insert

- Insert an unseen word into the dictionary.

Motivations

- We have a dictionary, e.g., {“a”, “absolute”, “accelerate”, …, “zoo”}.

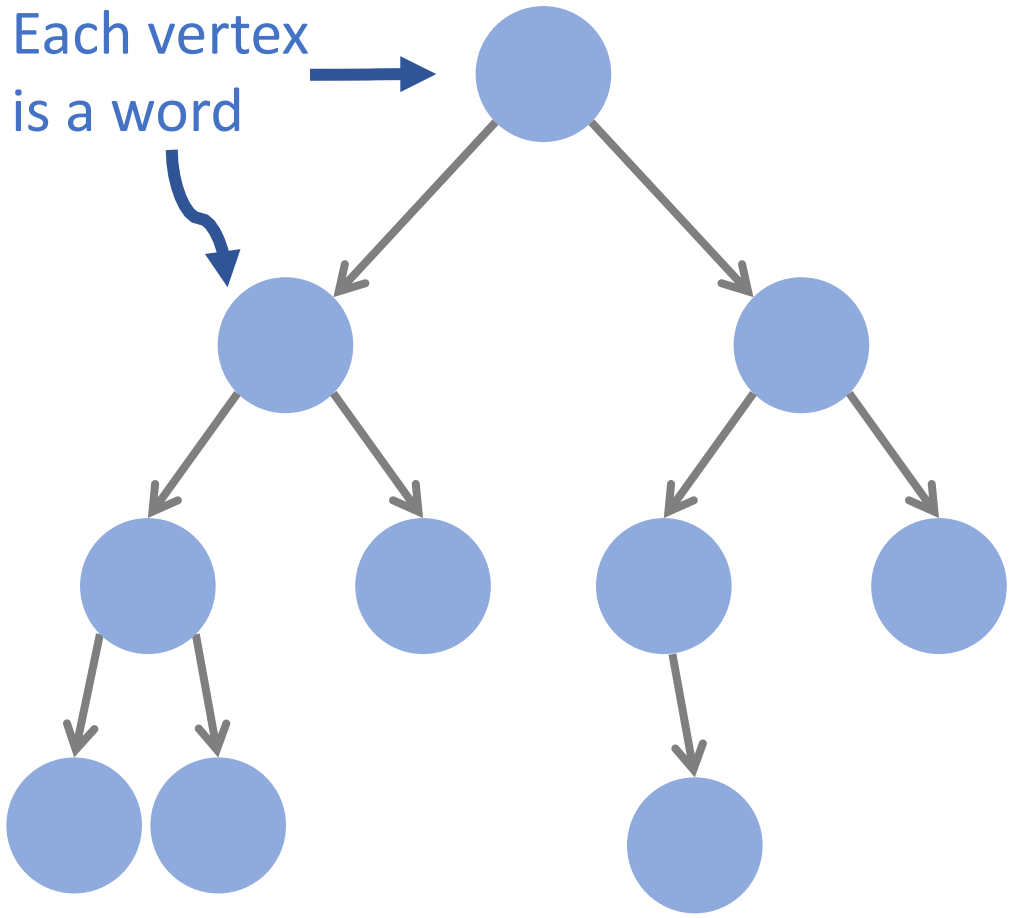
Task 1: Search

Task 2: Insert

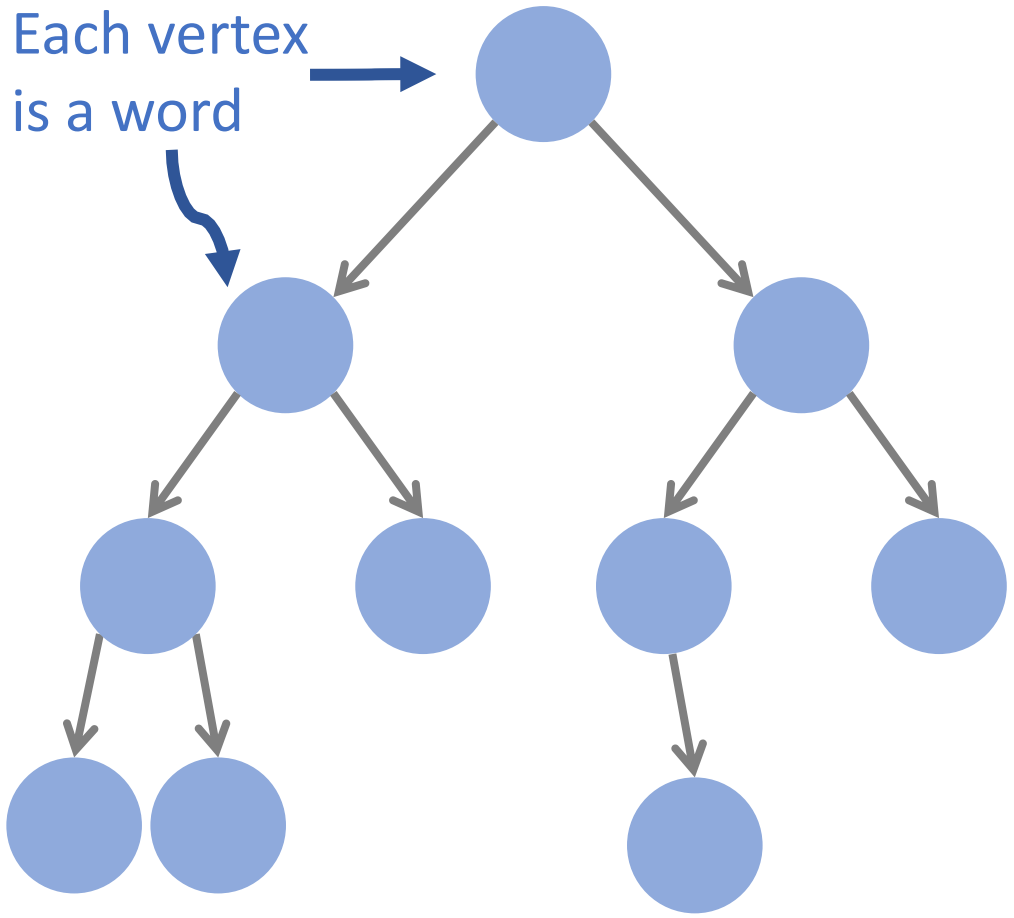
Task 3: Prefix Match

- Find all the words that start with some string.
- E.g., StartWith(“car”) = {“car”, “carbon”, “card”, “cargo”, “cart”, …}

Can we use binary search trees?



Can we use binary search trees?



- V: vocabulary (i.e., number of words in the dictionary.)
- At least $O(\log V)$ comparisons for finding the query word. (Because the depth of tree is $O(\log V)$.)
- Each comparison has $O(L)$ time complexity (L is the word length.)
- Overall time complexity of search:

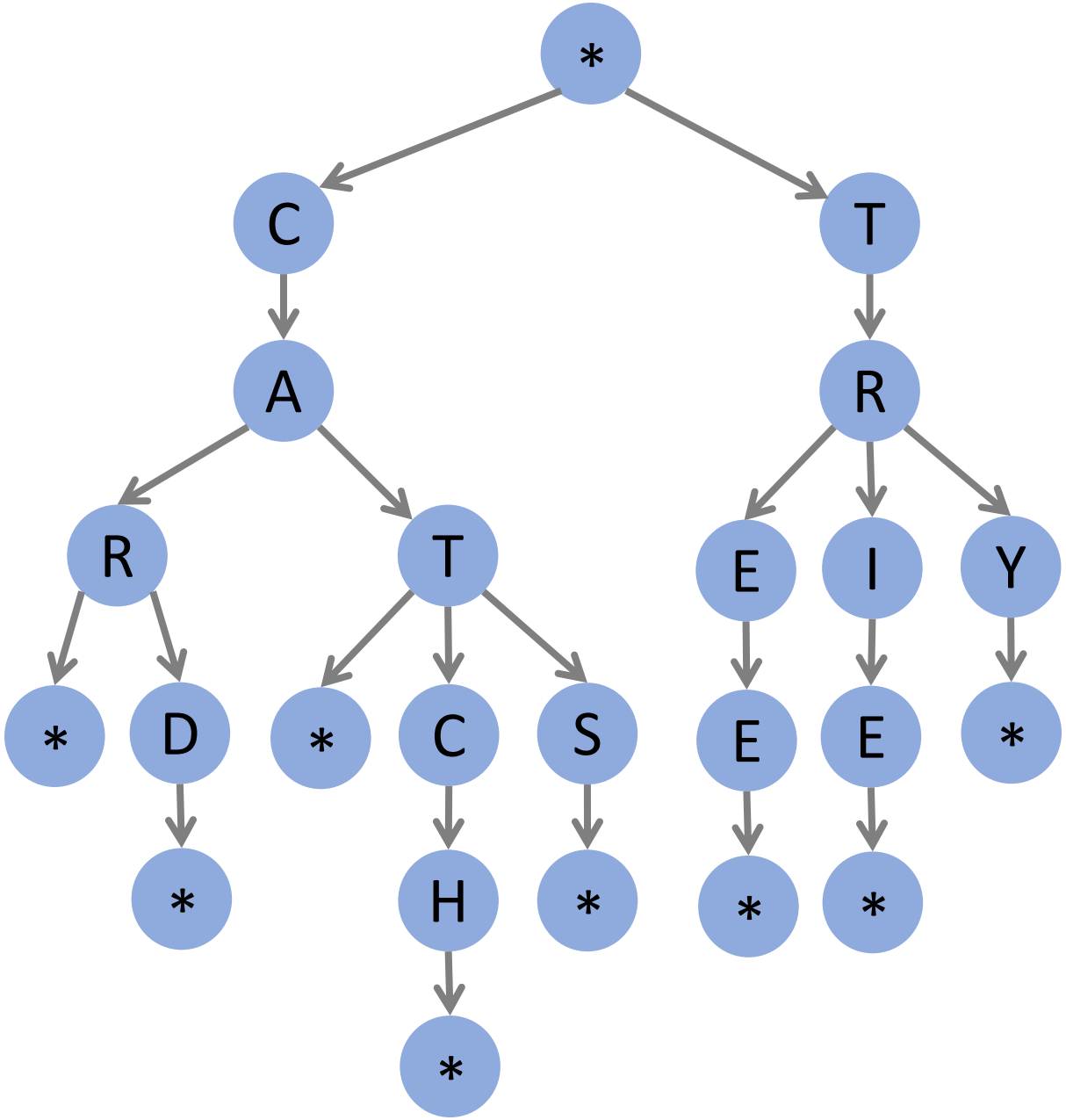
$$O(L \cdot \log V).$$

Can we use hash table?

Index:	1	…	13932	13933	13934	13935	…
Key:	“CART”	…	“TREE”		“CARGO”	“TRY”	…

- Near $O(1)$ comparisons for finding the query word.
- Each comparison has $O(L)$ time complexity (L is the word length.)
- Overall time complexity of search: $O(L)$.
- Hash table does not support **`startWith()`**.

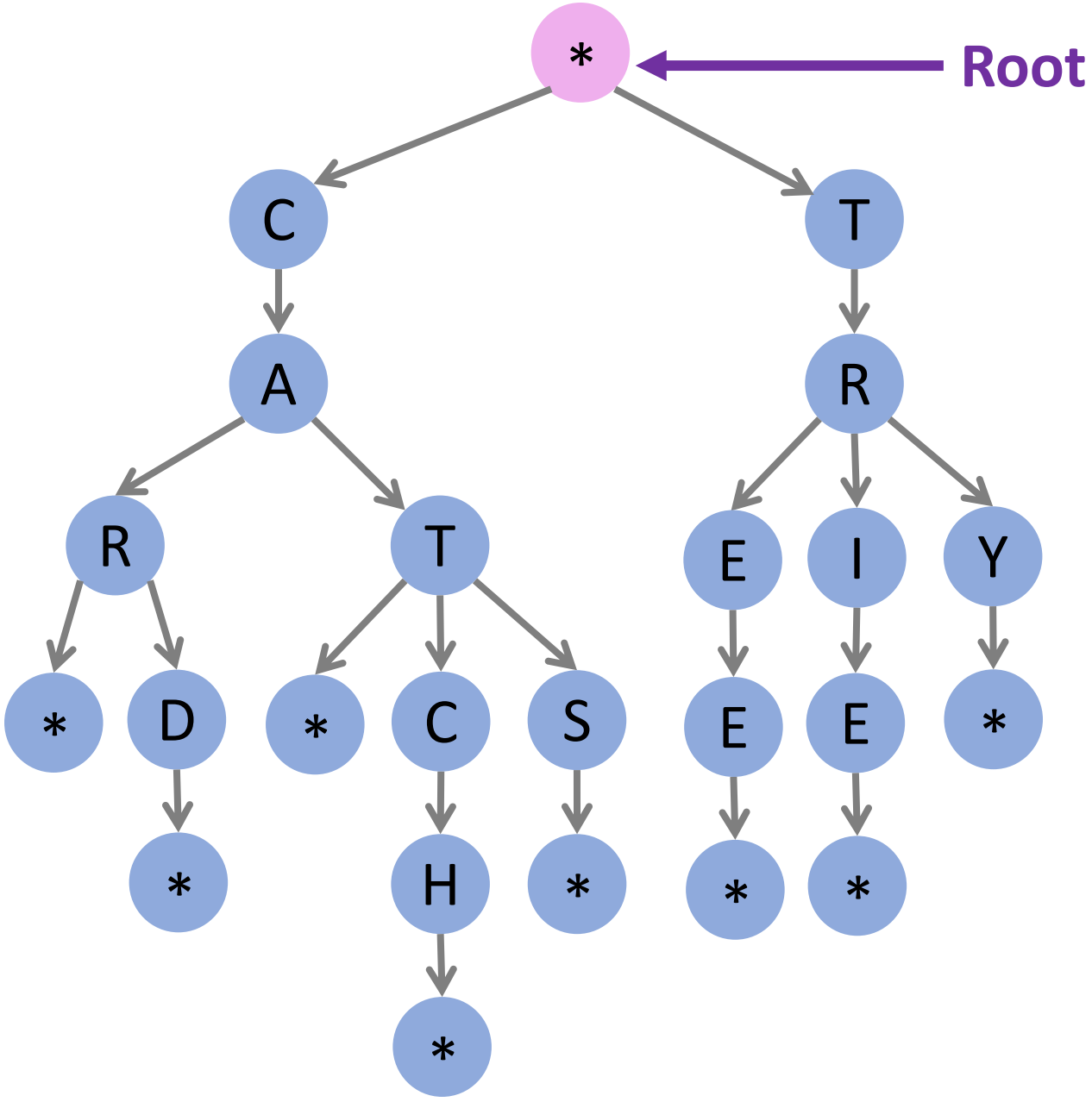
Trie Data Structure

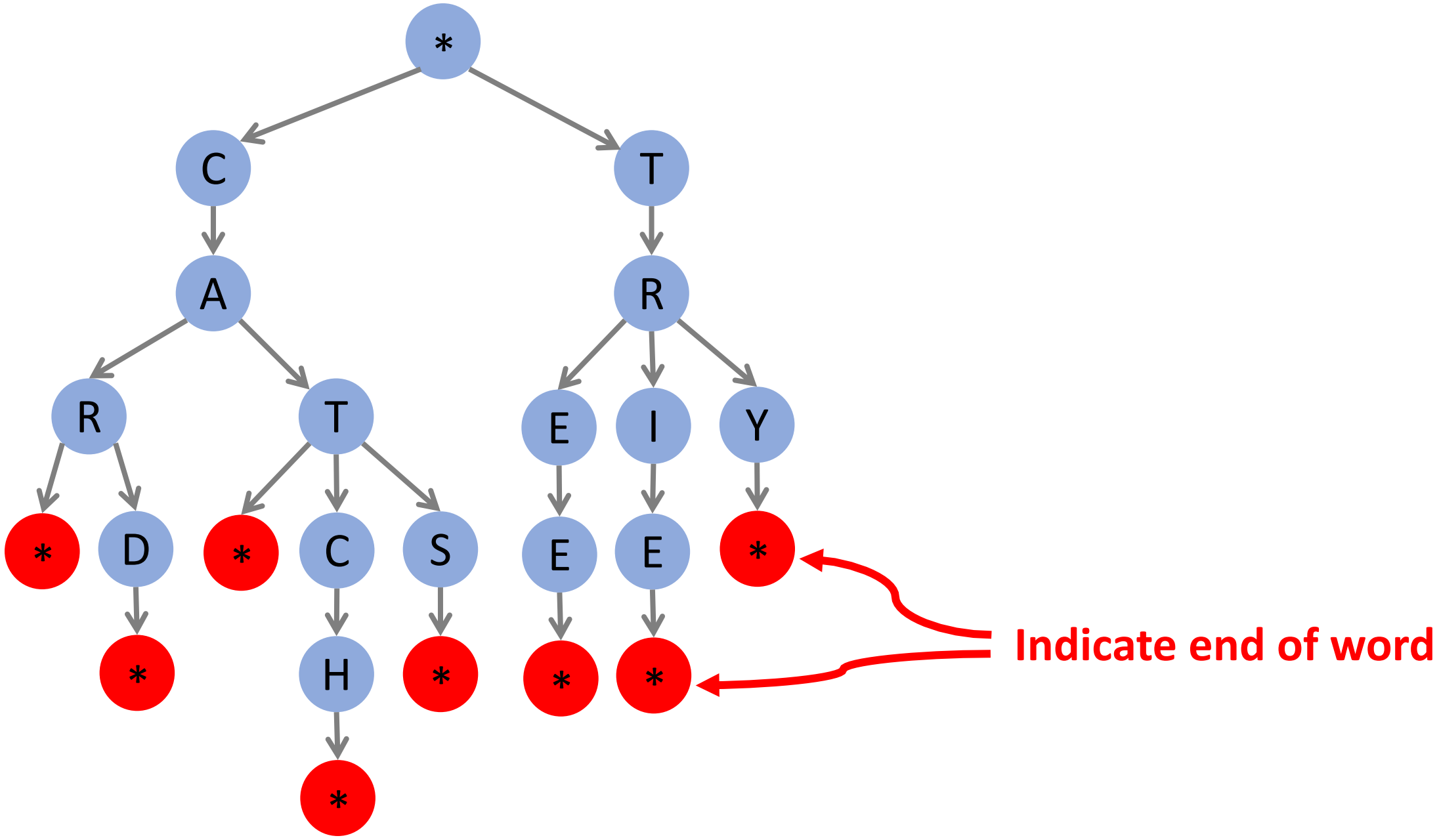


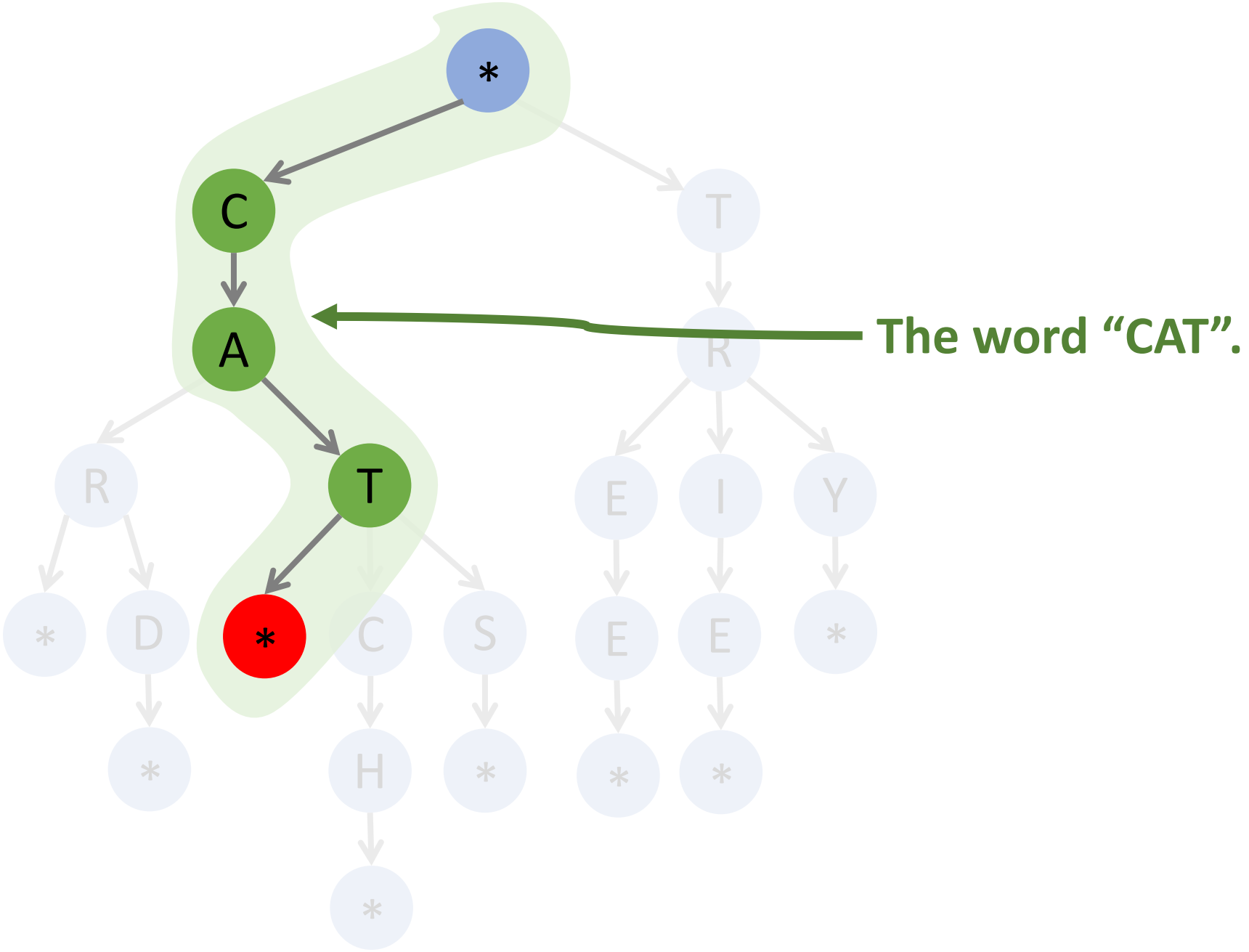
Trie

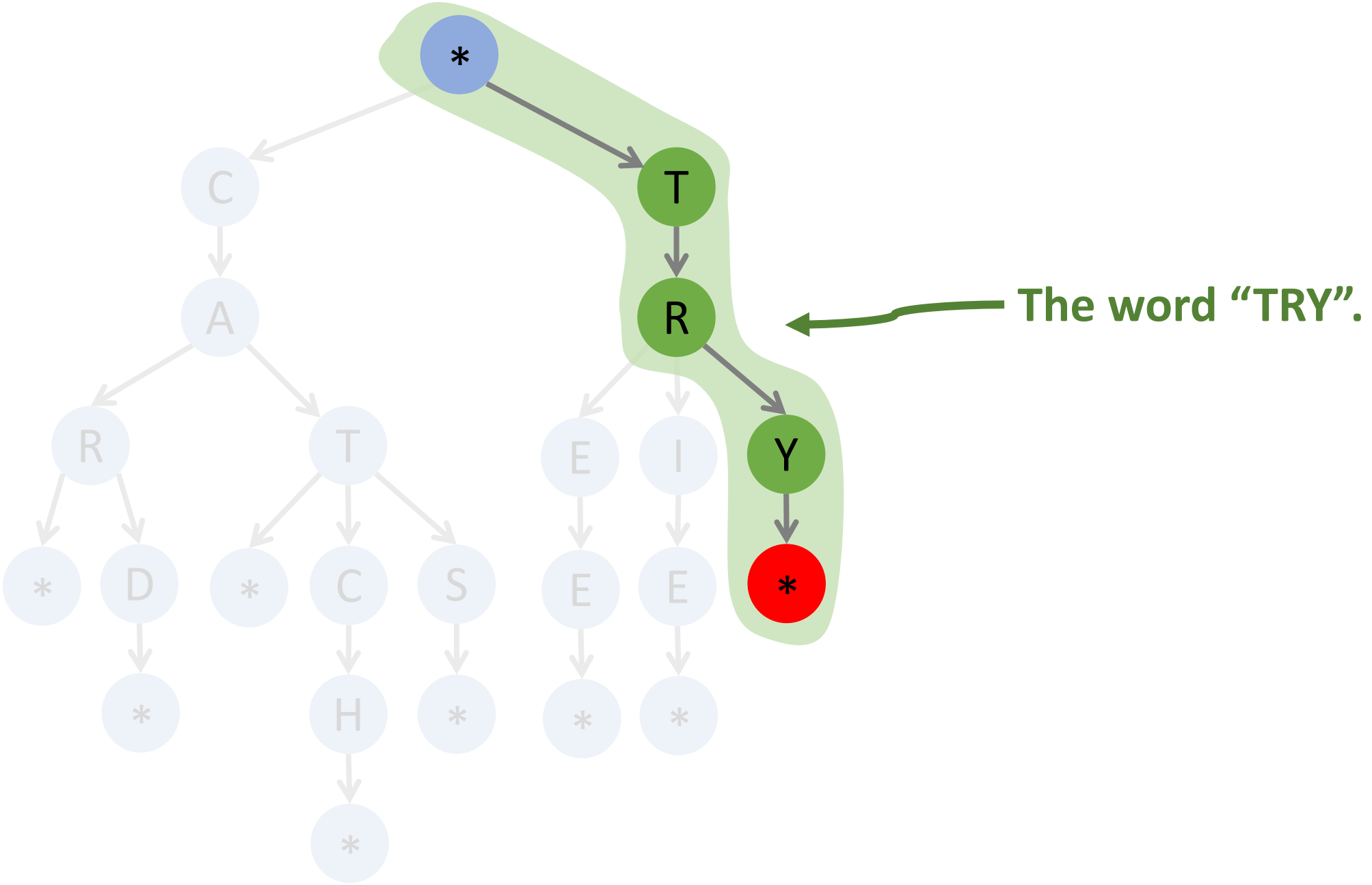
- **Trie** is also called **prefix tree** or **digital tree**.
- It supports
  - **search** in $O(L)$ time,
  - **insert** in $O(L)$ time,
  - **startWith** in $O(L)$ time,

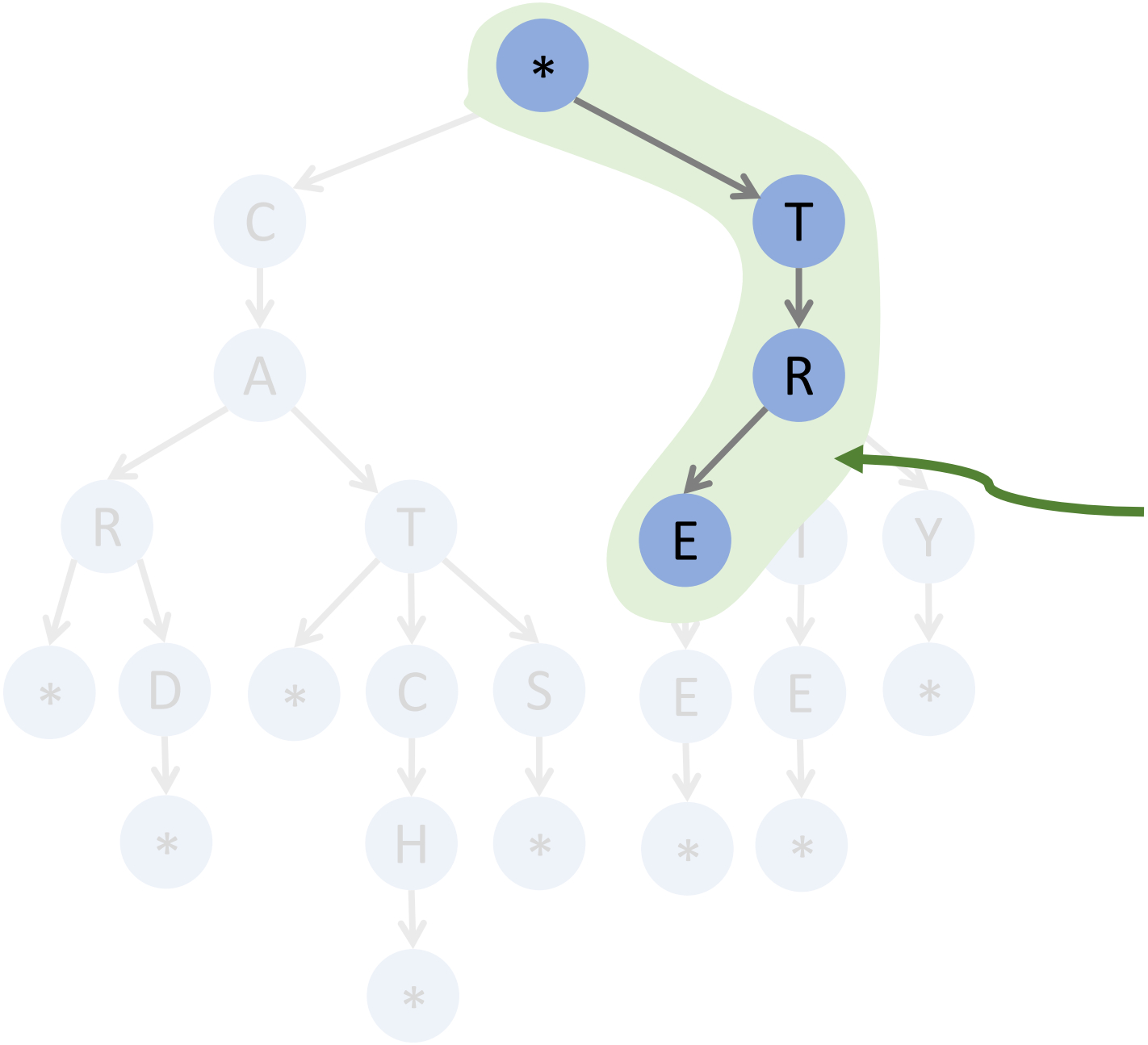
  where L is query's length.


*
Root
C
T
A
R
R
T
E
I
Y
*
D
*
C
S
E
E
*
*
*
H
*
*
*
*

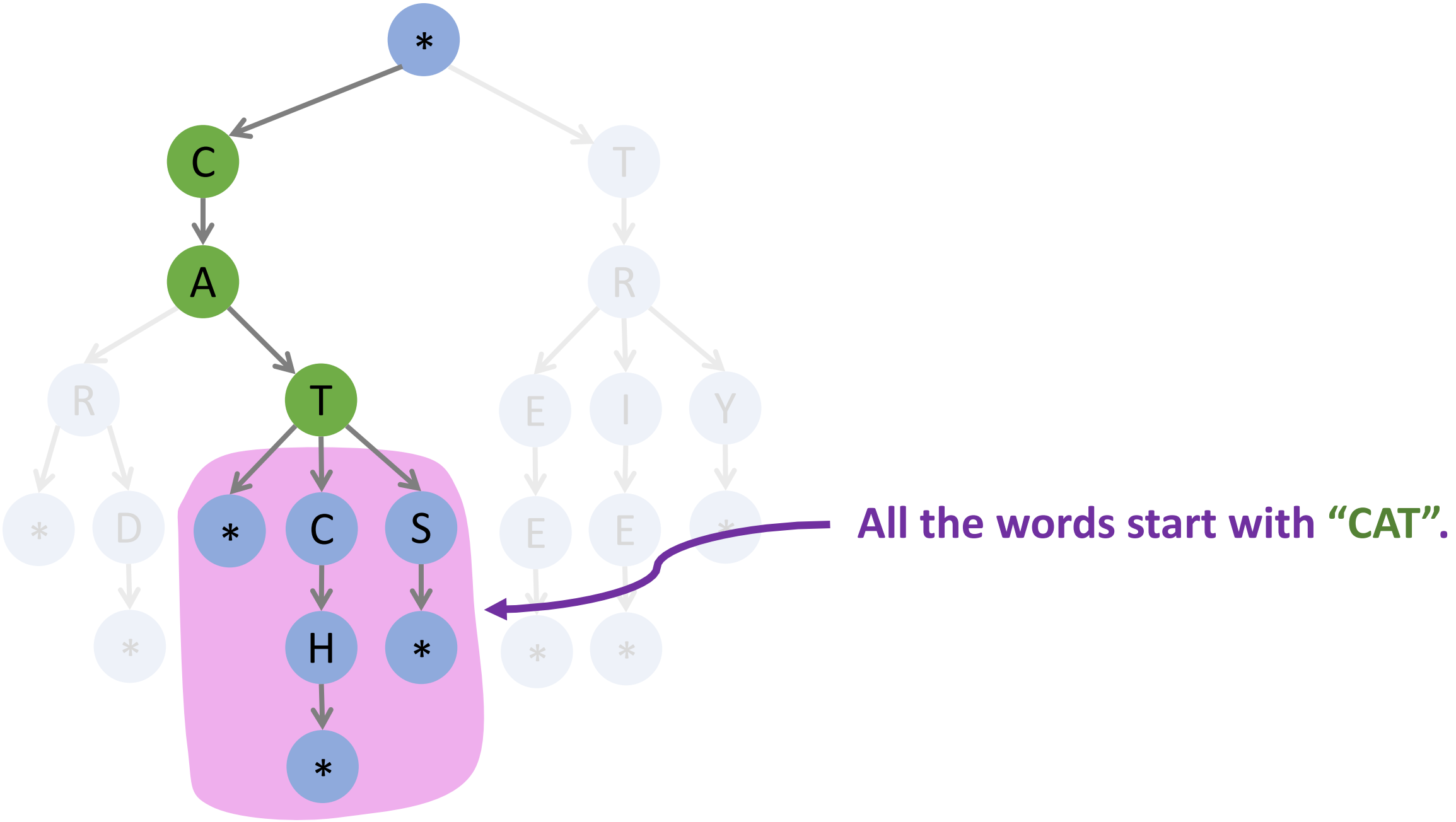

*
C
T
A
R
R
T
E
I
Y
*
D
*
C
S
E
E
*
*
H
*
*
*
*
Indicate end of word

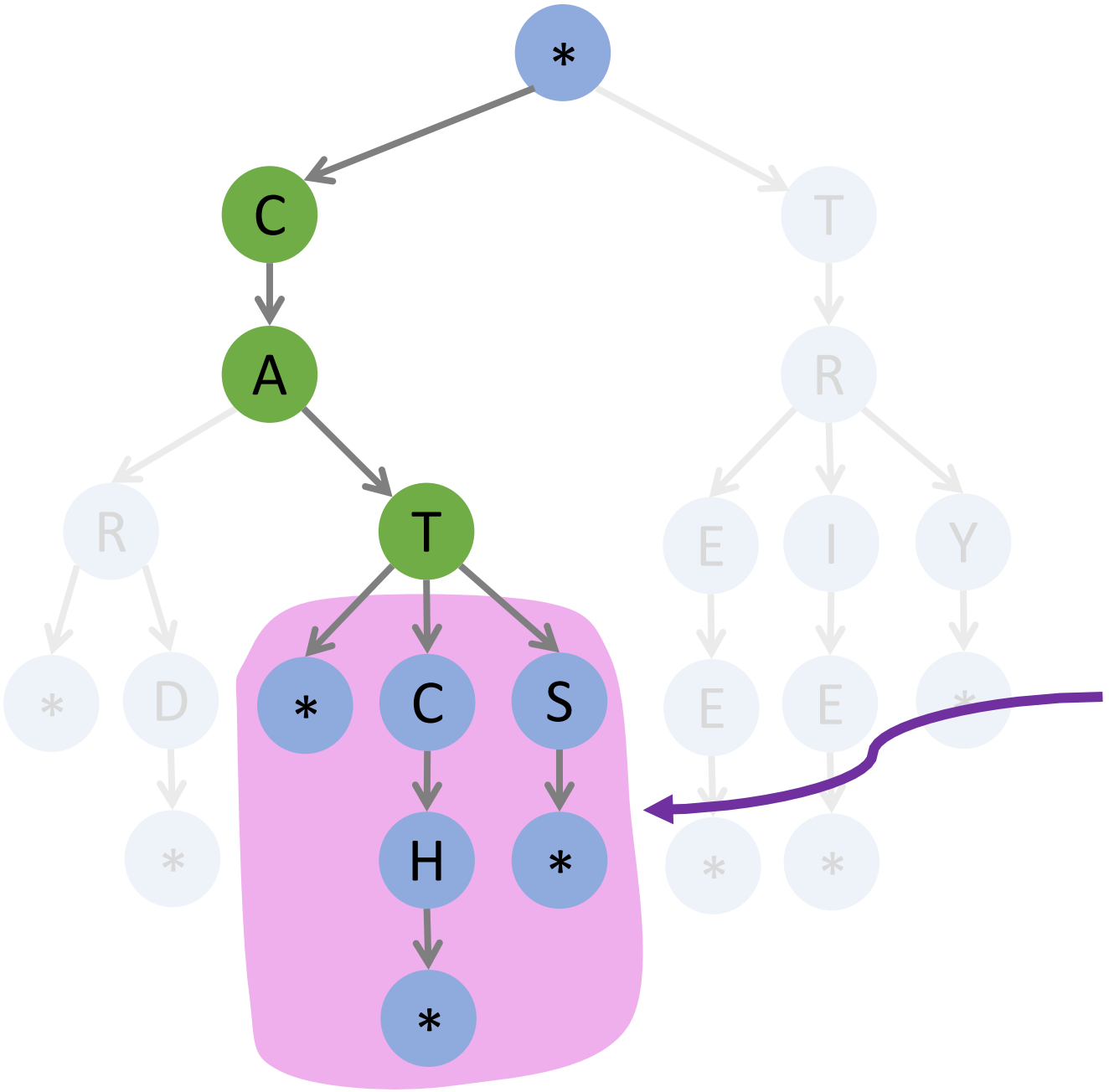

*
C
A
T
*
T
R
R
E
I
Y
*
D
C
S
E
E
*
*
H
*
*
*
*
The word “CAT”.


*
C
T
A
R
R
T
E
I
Y
*
D
*
C
S
E
E
*
H
*
*
*
*
*
The word “TRY”.



- **Not a word!**
- **Node ‘E’ does not have child ‘*’.**


*
C
A
T
R
T
R
E
I
Y
*
D
*
C
S
H
*
*
E
E
*
*
*
All the words start with “CAT”.

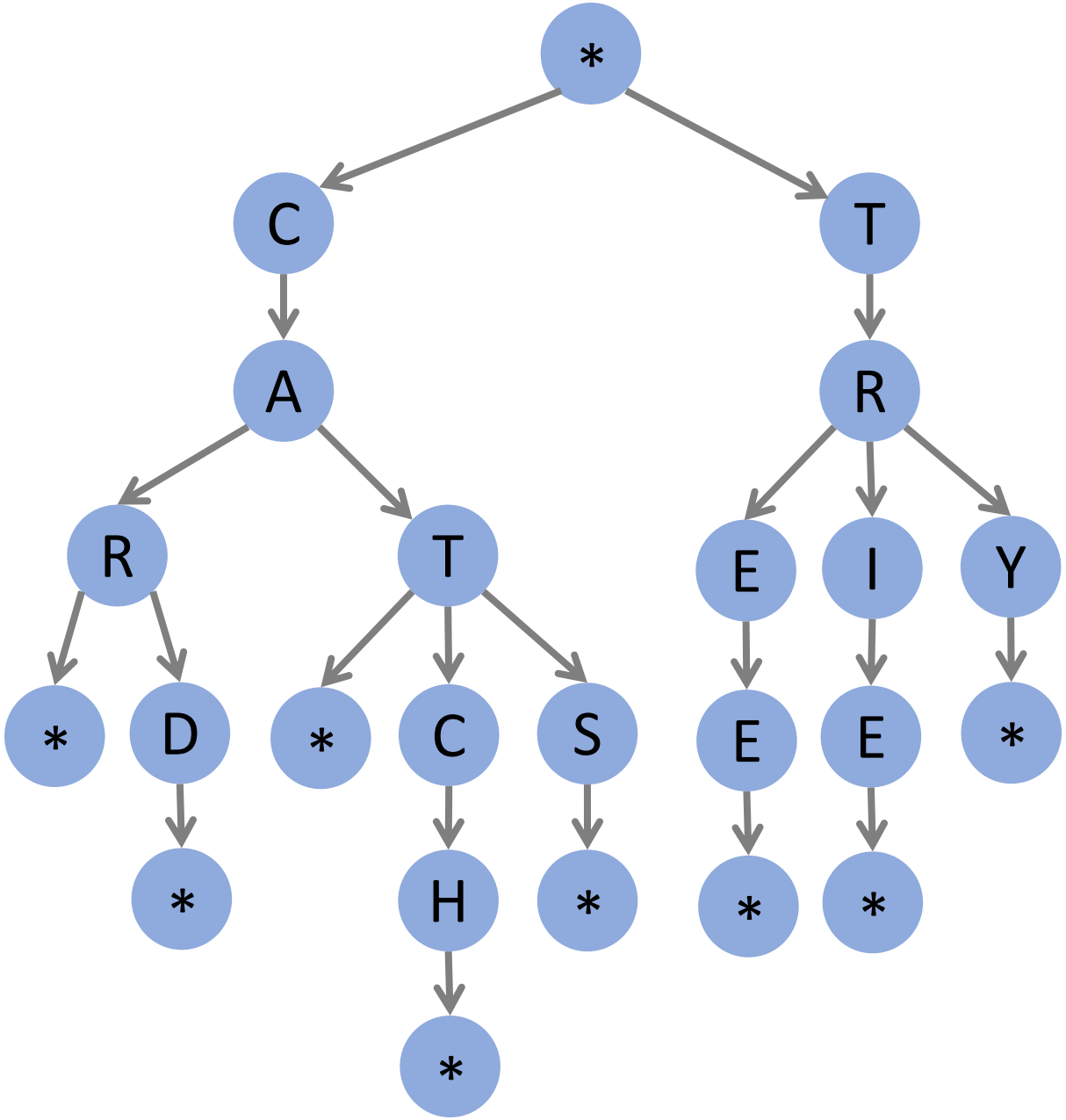


Words that starts with “CAT”:

- “CAT”,
- “CATCH”,
- “CATS”.

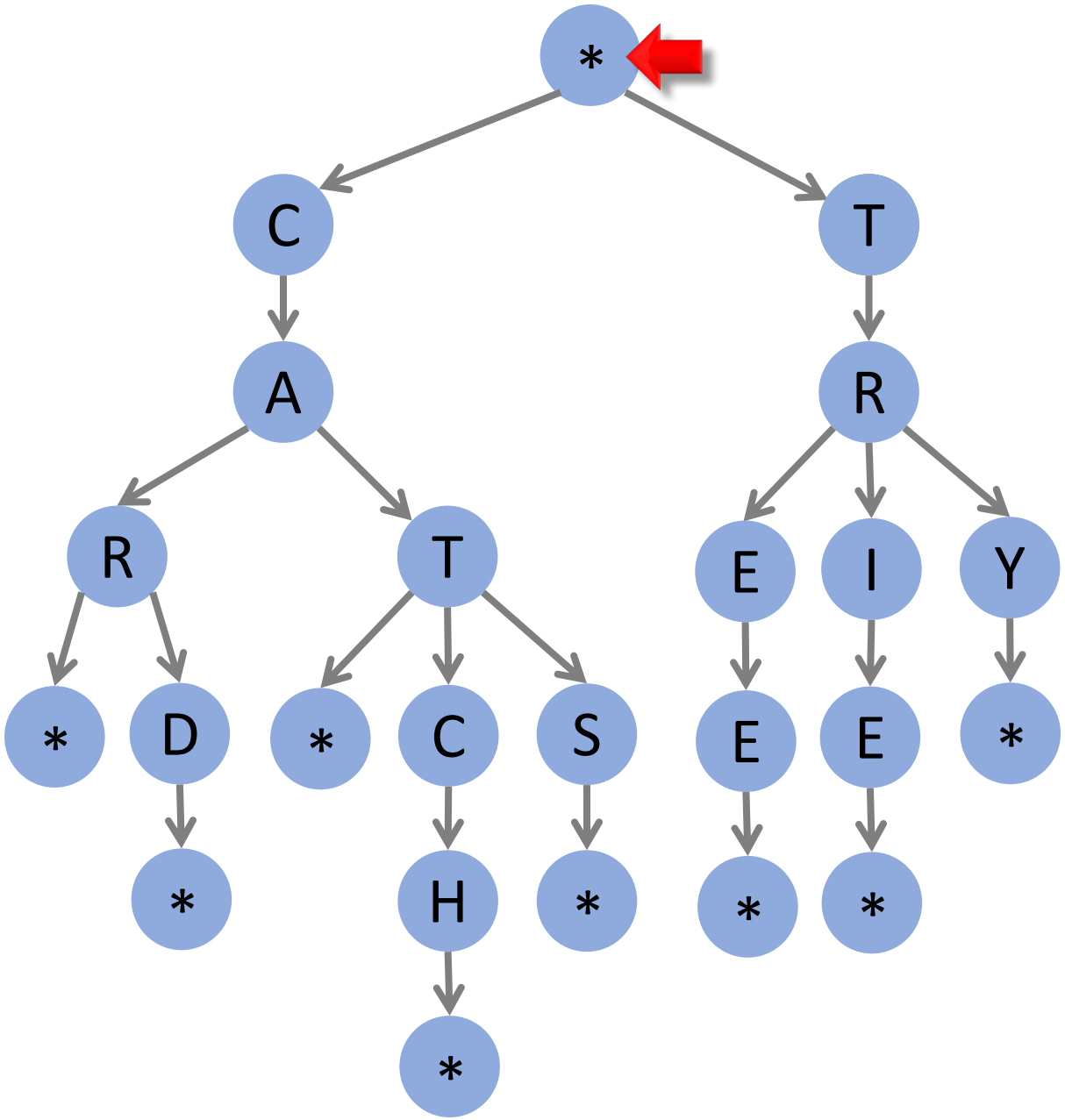
Task 1: Search

Search: Example 1



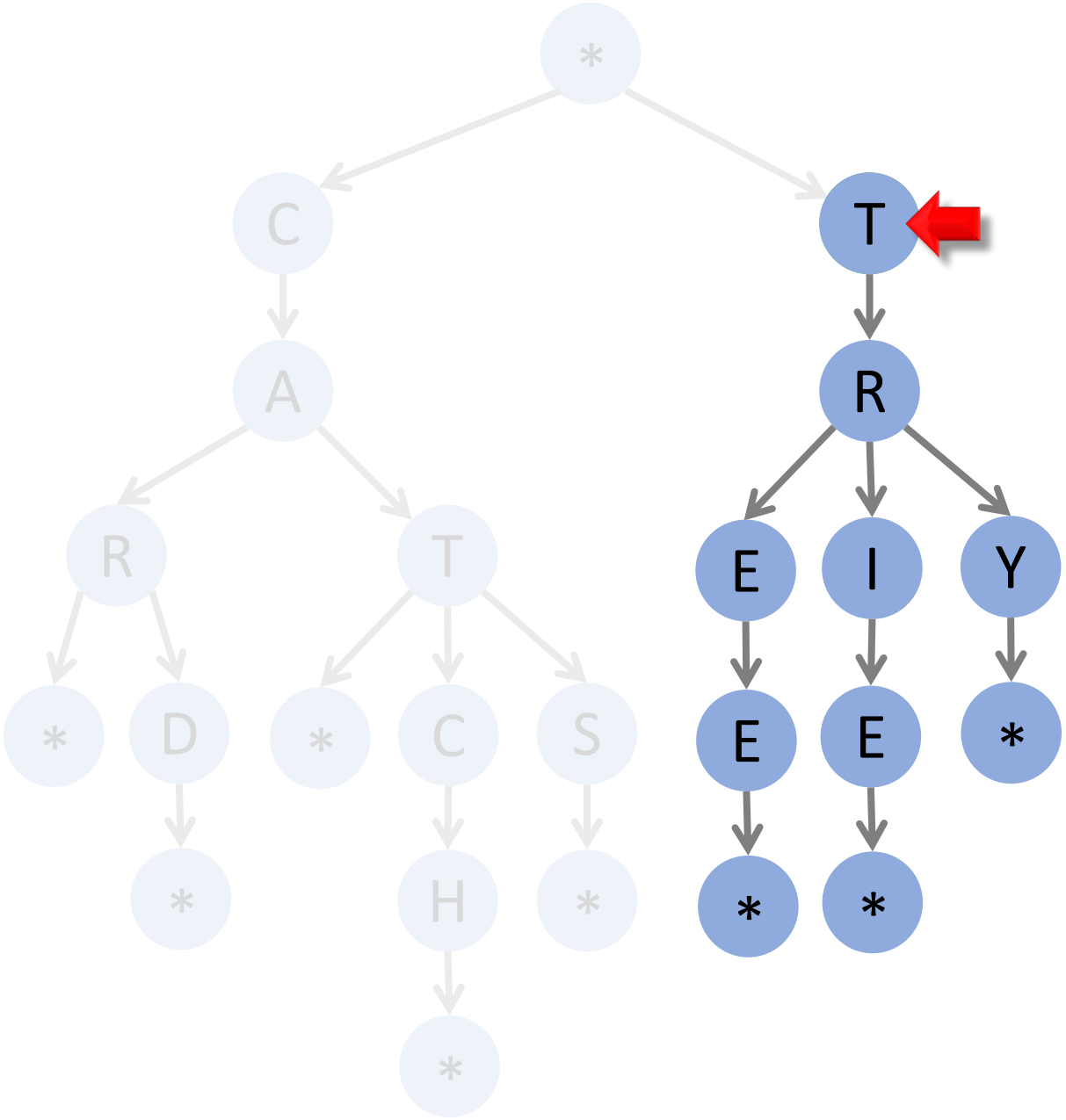
Query: “ T R E E ”

Search: Example 1



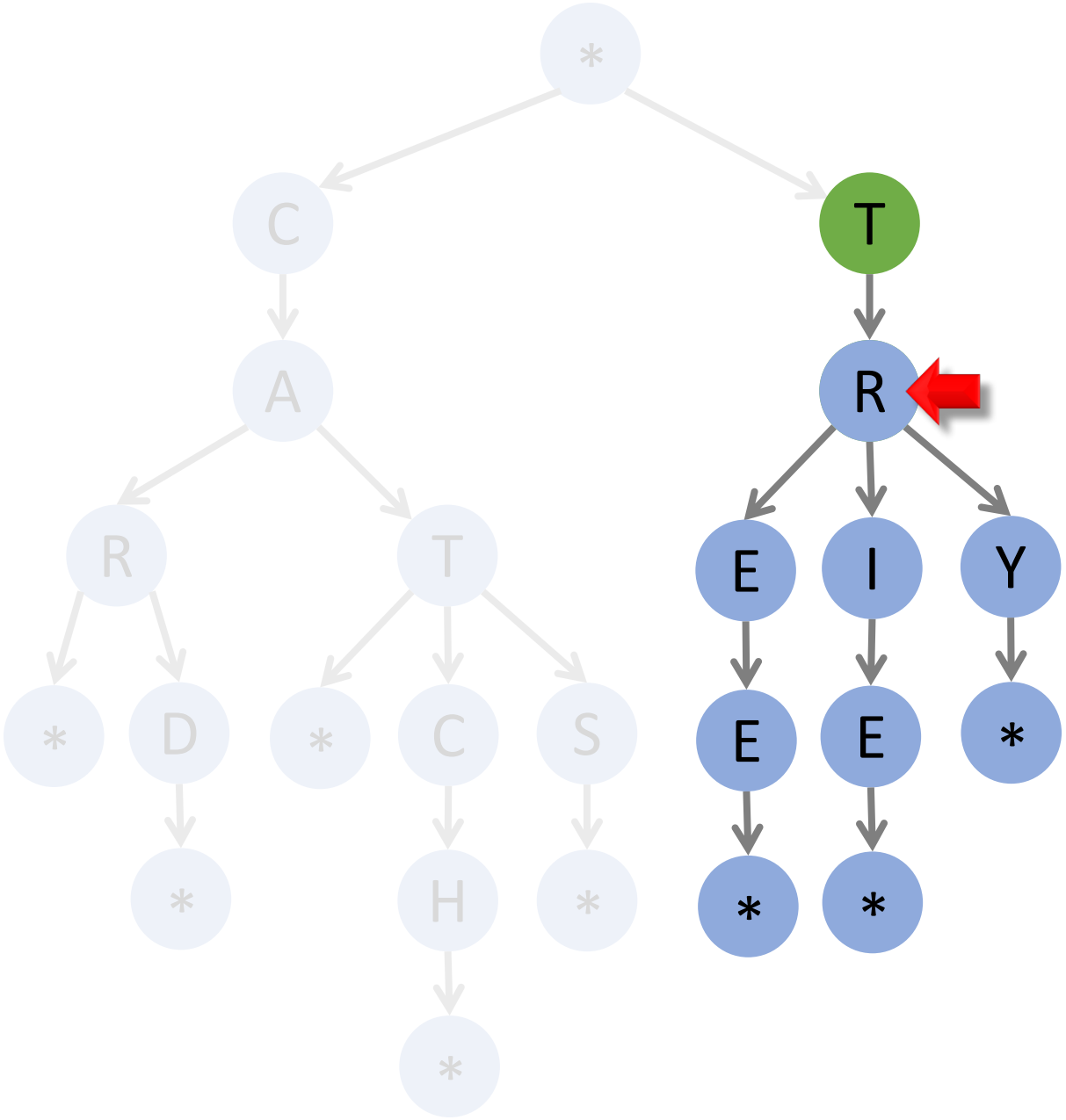
Query: “ T R E E ”

Search: Example 1



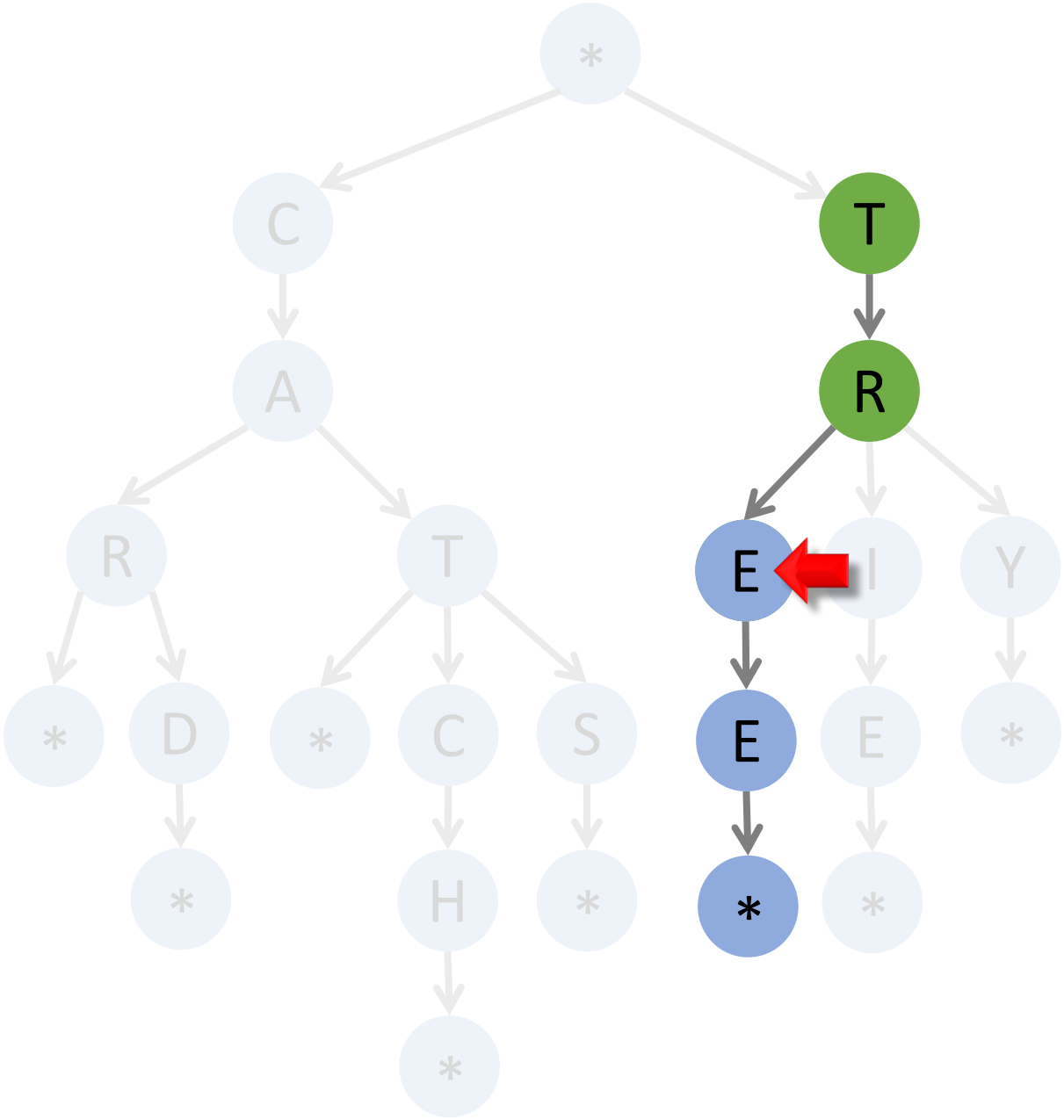
Query: “ T R E E ”

Search: Example 1



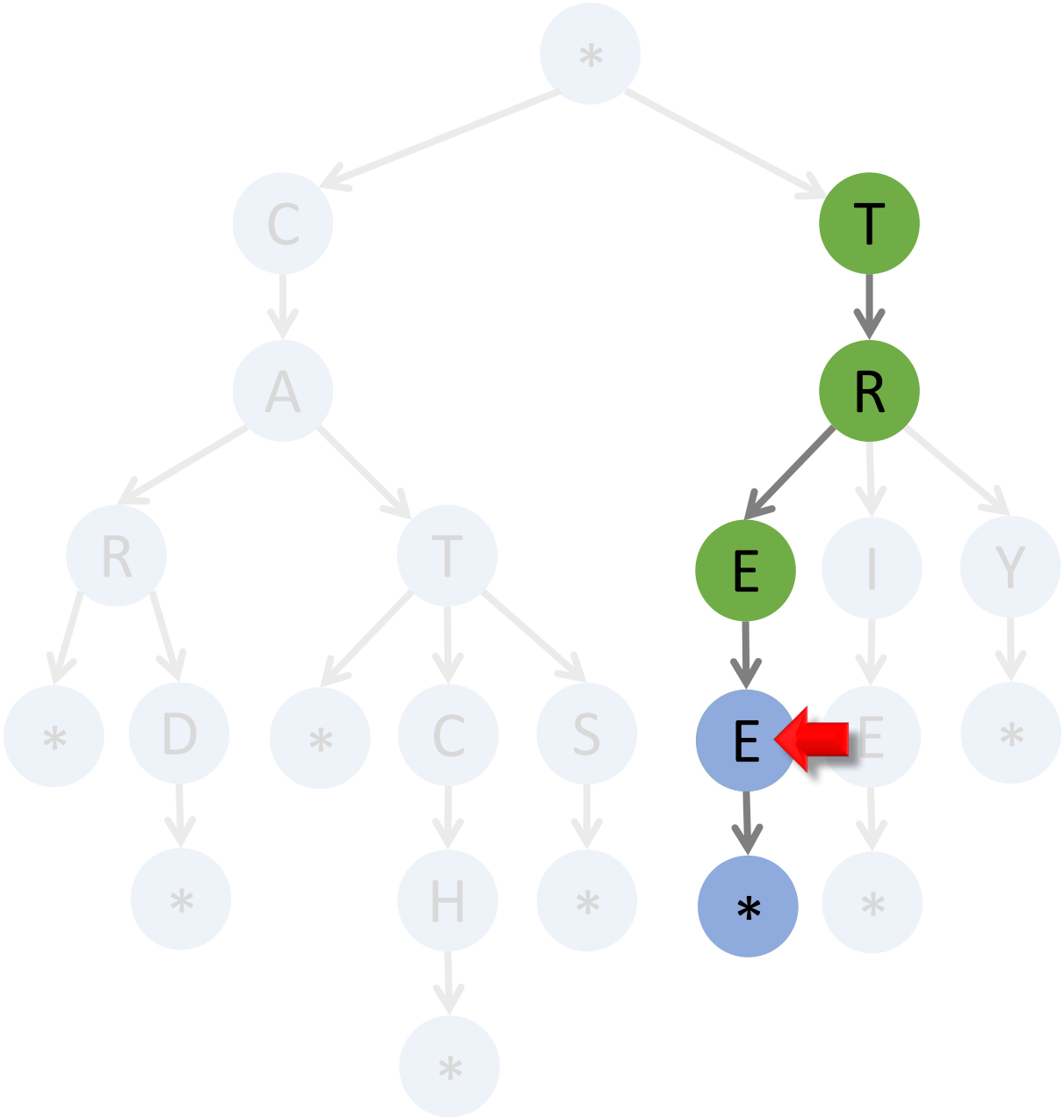
Query: " T R E E "

Search: Example 1



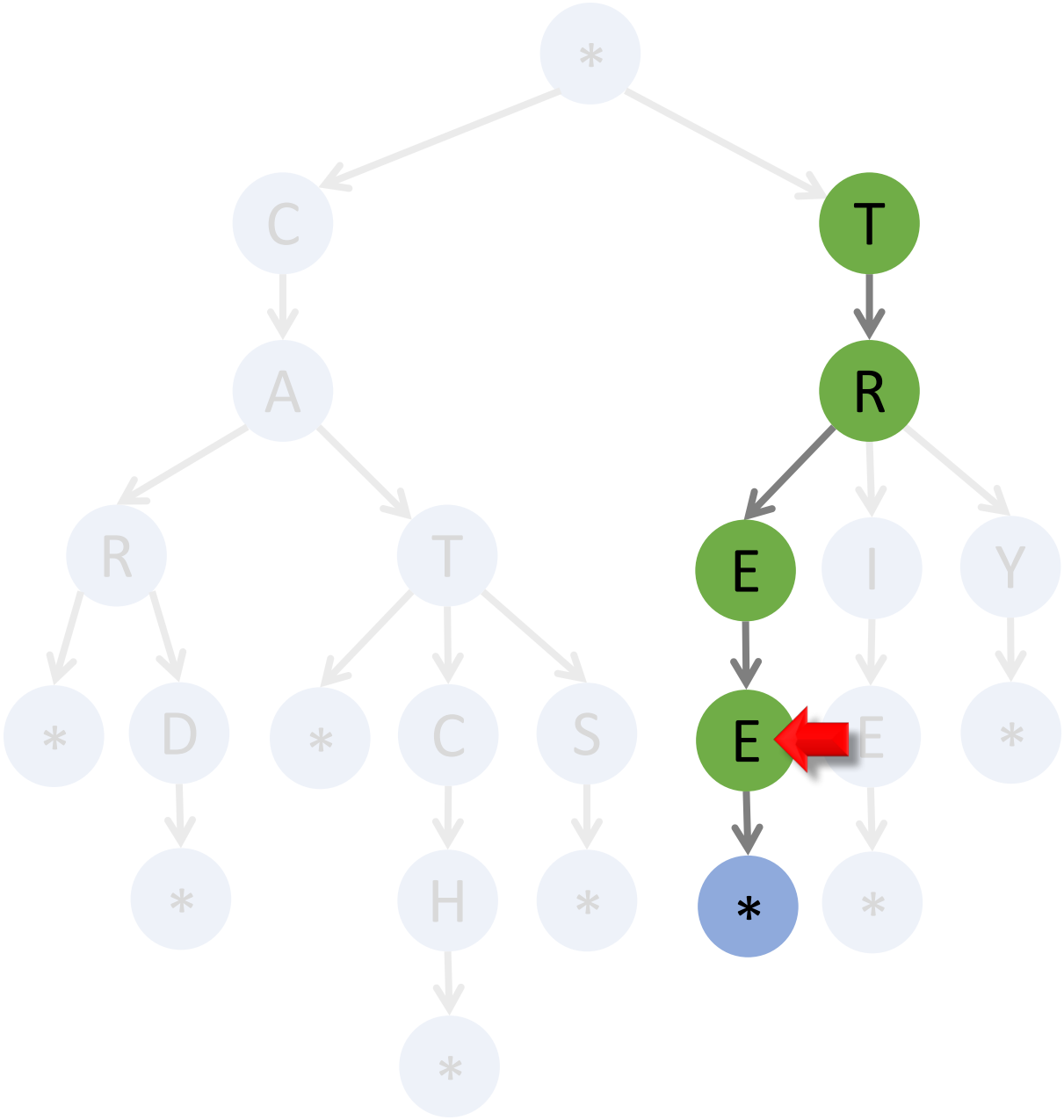
Query: “ T R E E ”

Search: Example 1



Query: “ T R E E ”

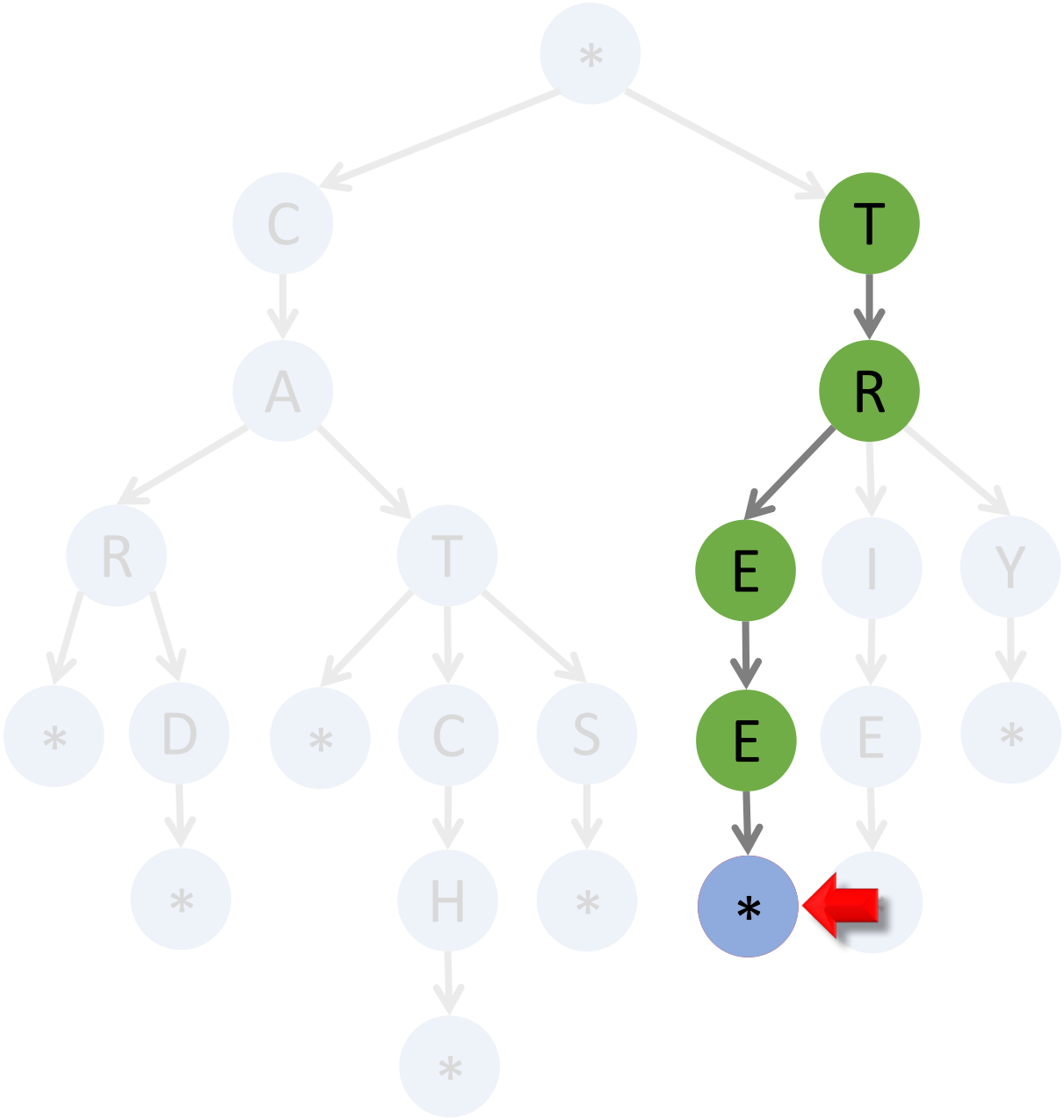
Search: Example 1



Query: “ T R E E ”

- ‘E’ is the end of word.
- Does ‘E’ have a child of ‘*’?

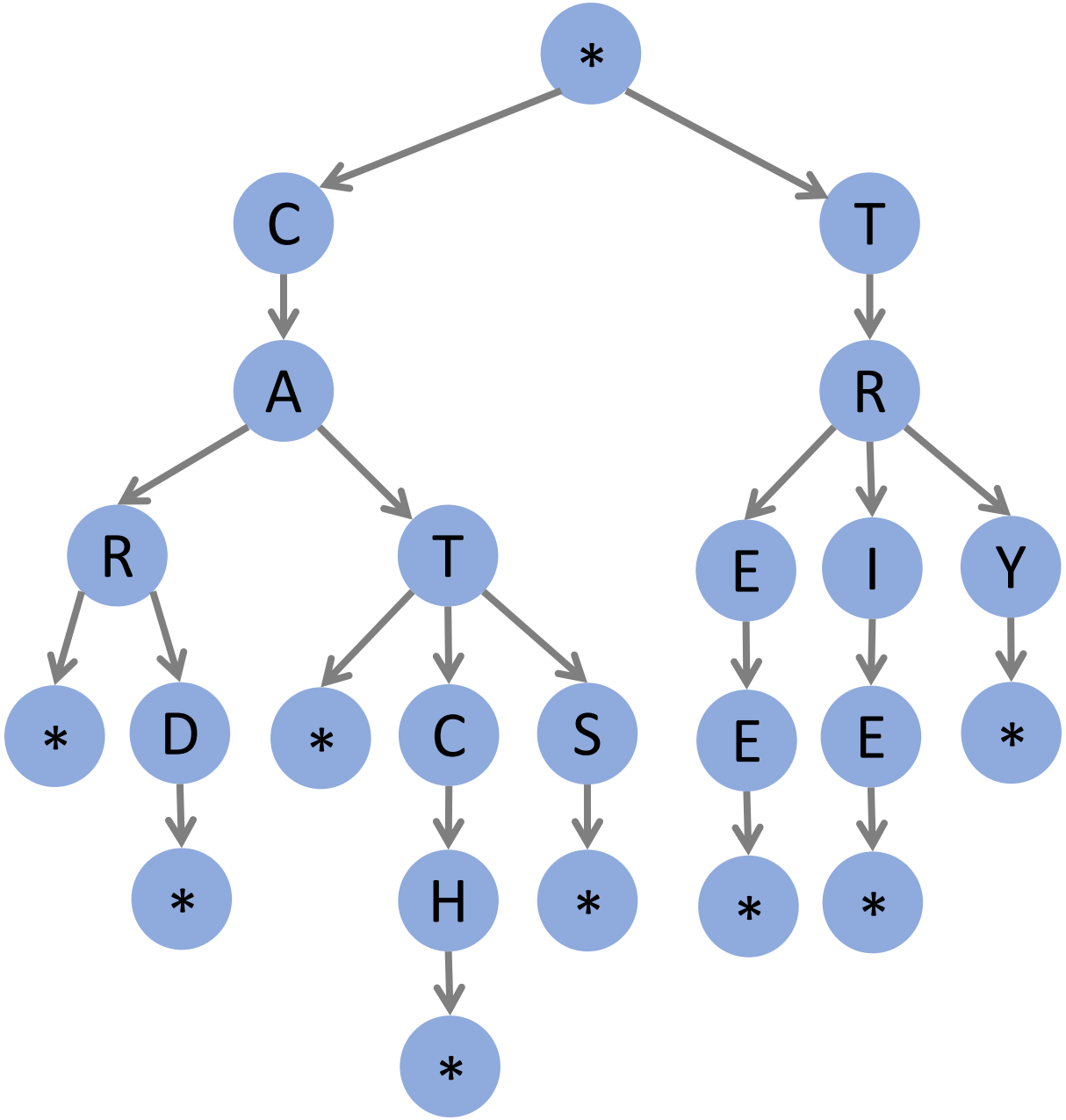
Search: Example 1



Query: “ T R E E ”

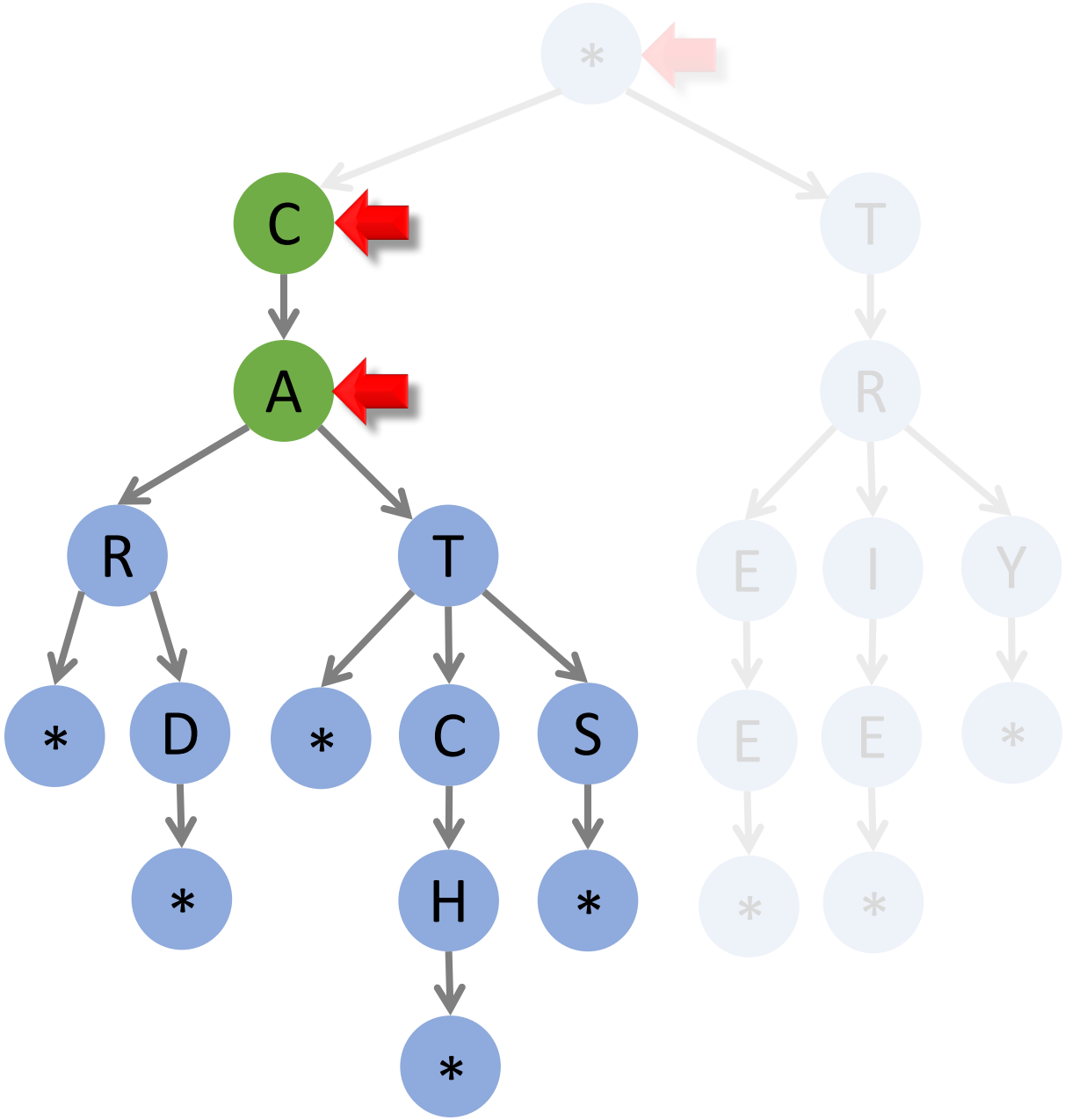
- ‘E’ is the end of word.
- Does ‘E’ have a child of ‘*’?
- Yes ➔ “TREE” is in the dictionary.

Search: Example 2



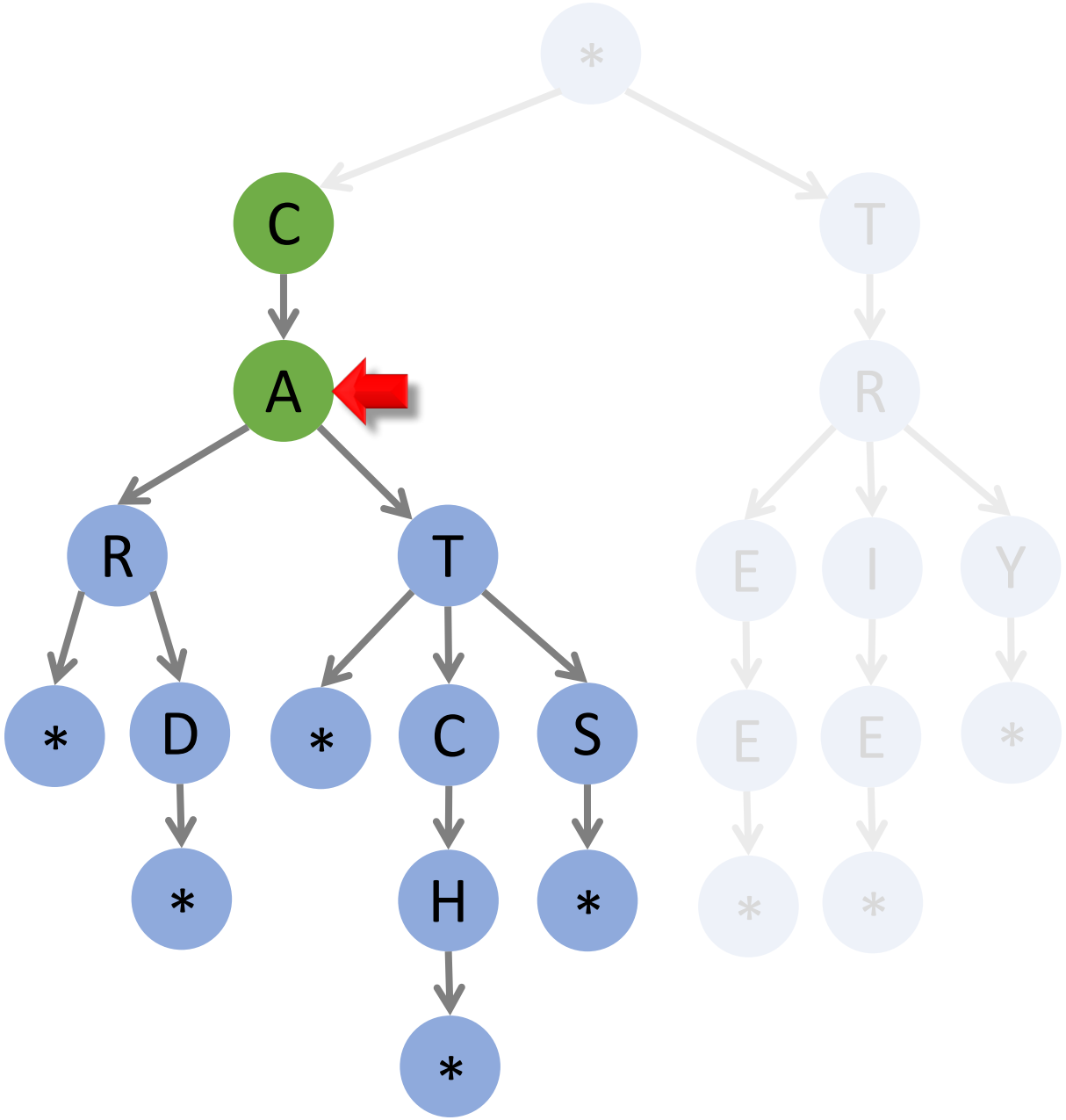
Query: “ C A ”

Search: Example 2



Query: “ C A ”

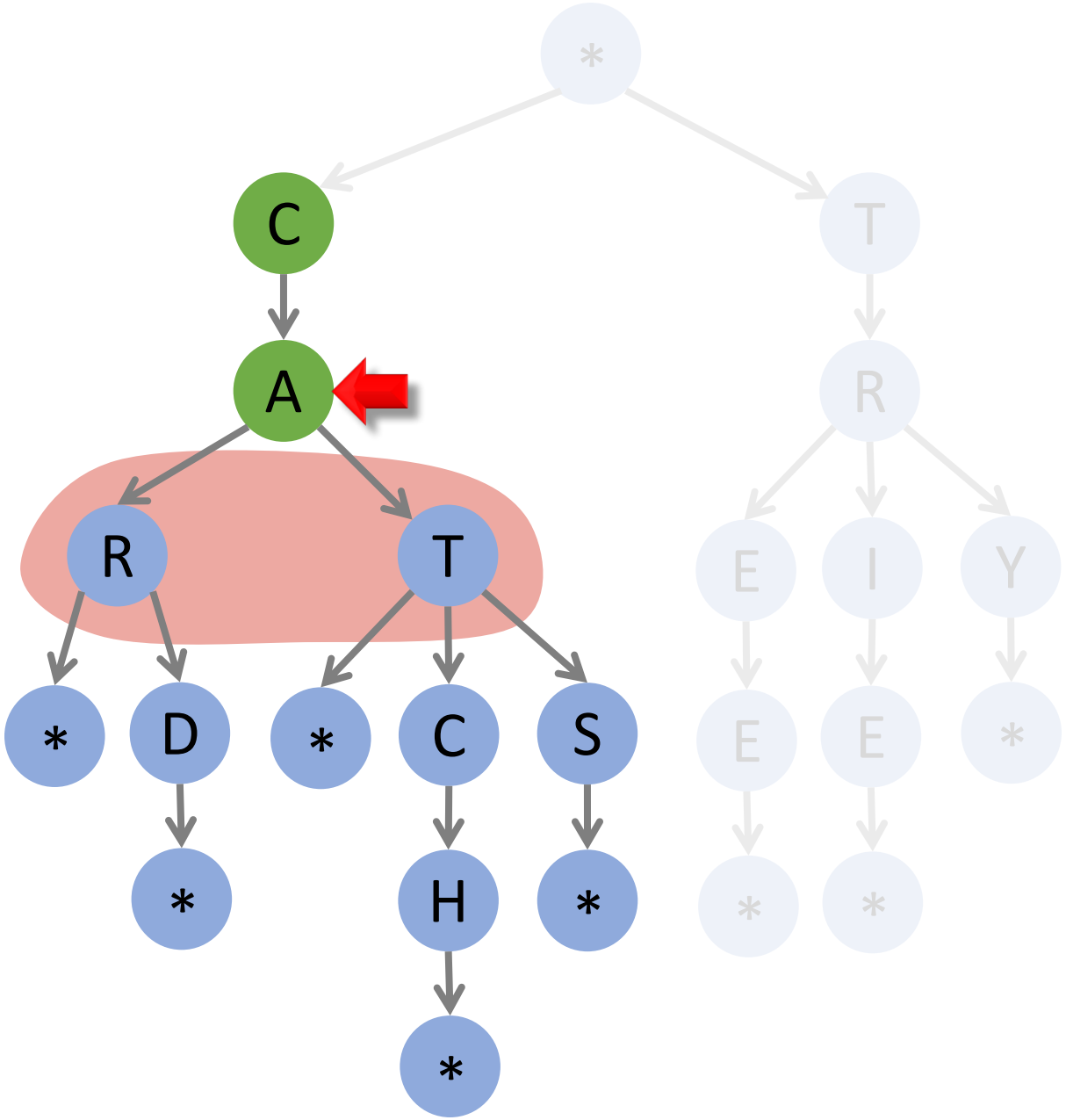
Search: Example 2



Query: “ C A ”

- ‘A’ is the end of word.
- Does ‘A’ have a child of ‘*’?

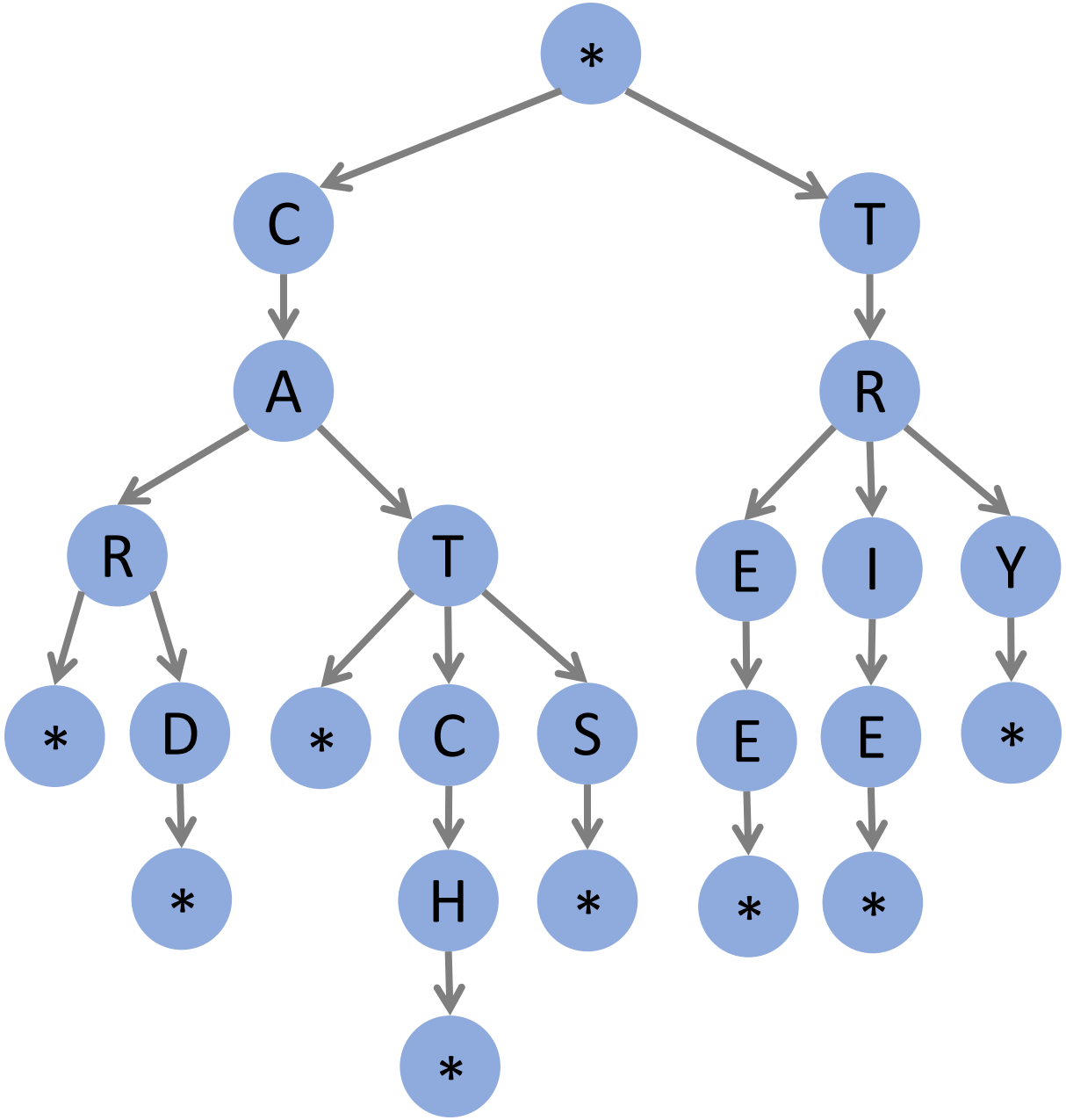
Search: Example 2



Query: “ C A ”

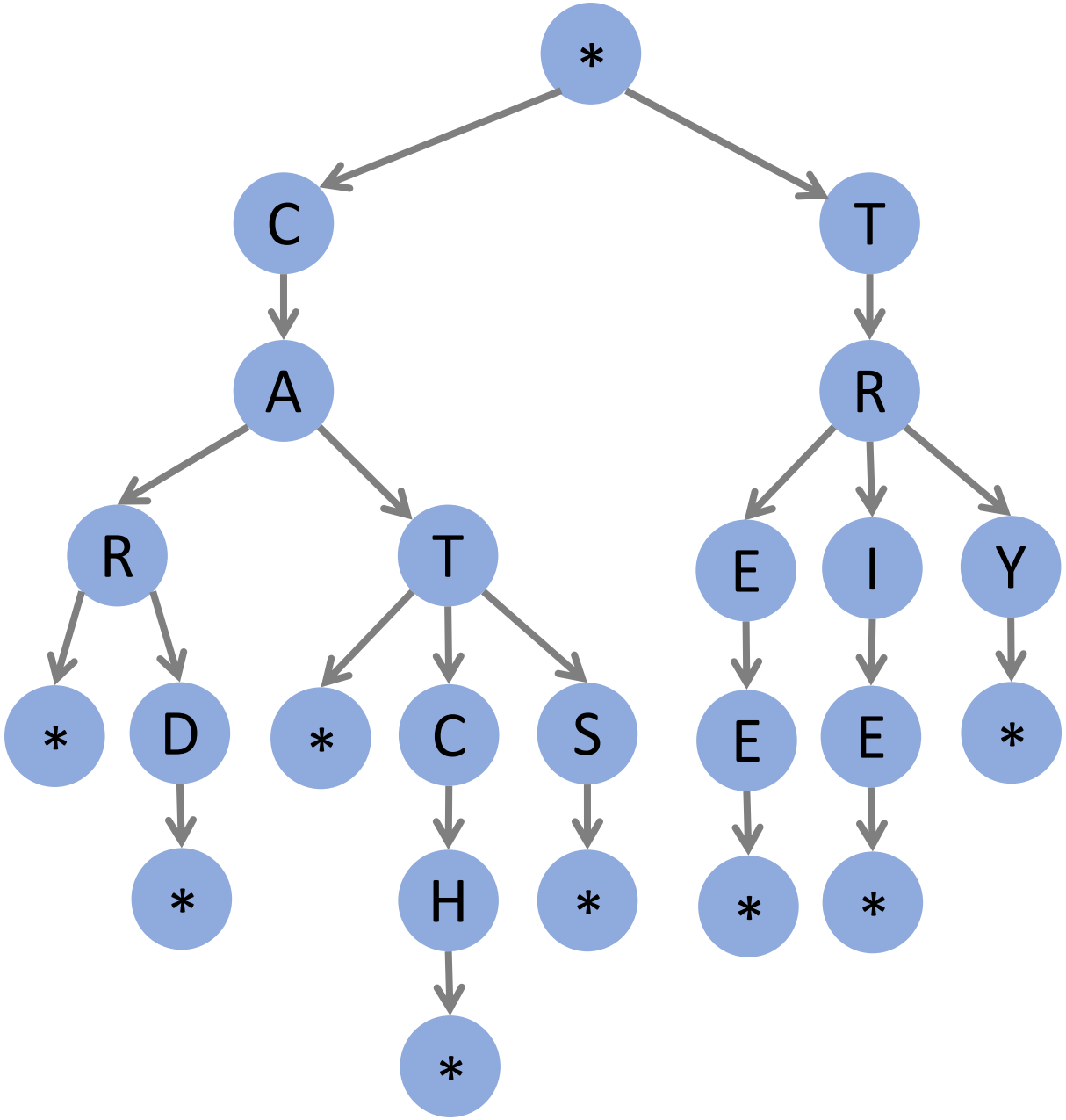
- ‘A’ is the end of word.
- Does ‘A’ have a child of ‘*’?
- No ➔ “CA” is in the dictionary.

Search: Example 3



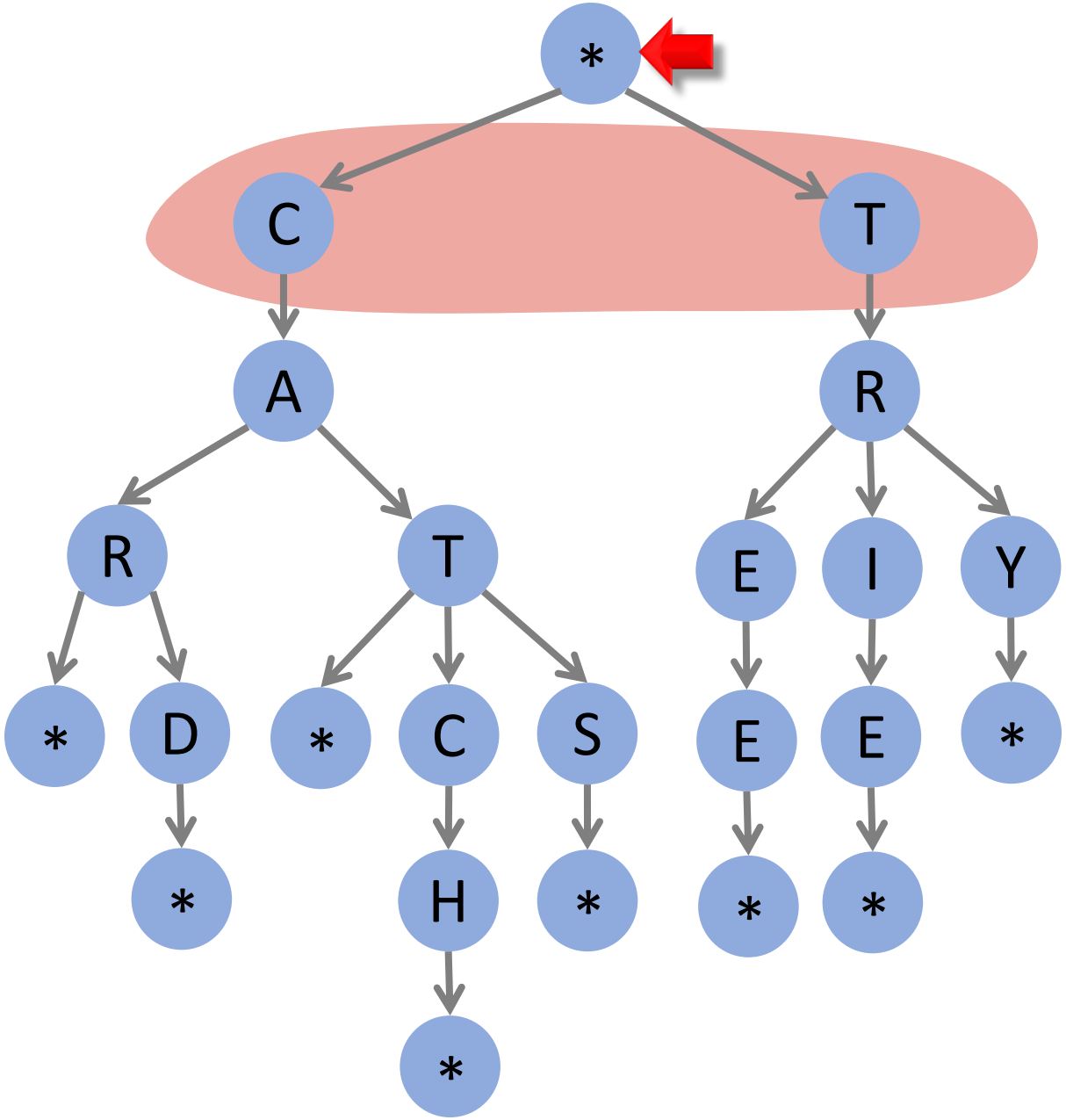
Query: “ H E A P ”

Search: Example 3



Query: “ H E A P ”

Search: Example 3

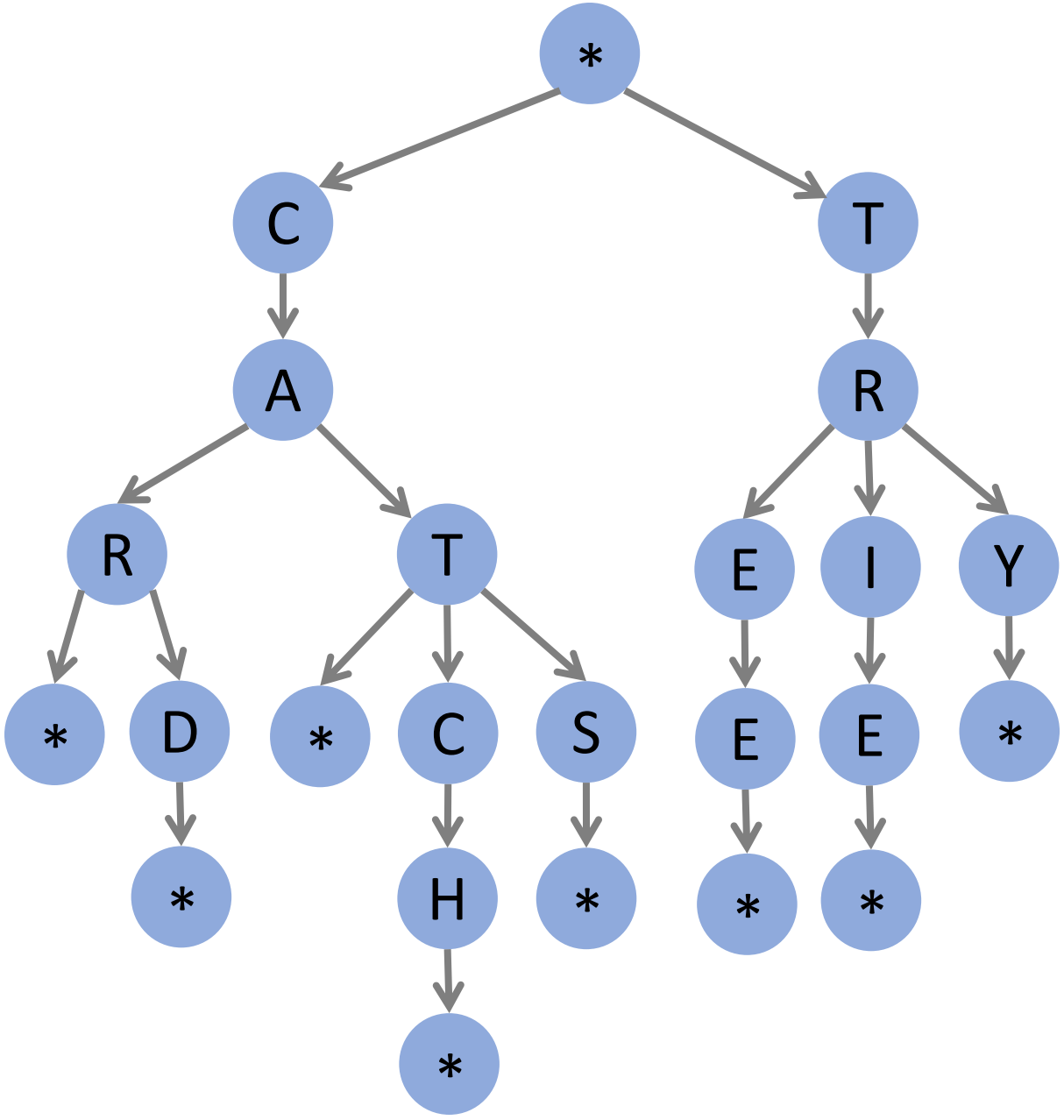


Query: “ H E A P ”

- “HEAP” is not in the dictionary.

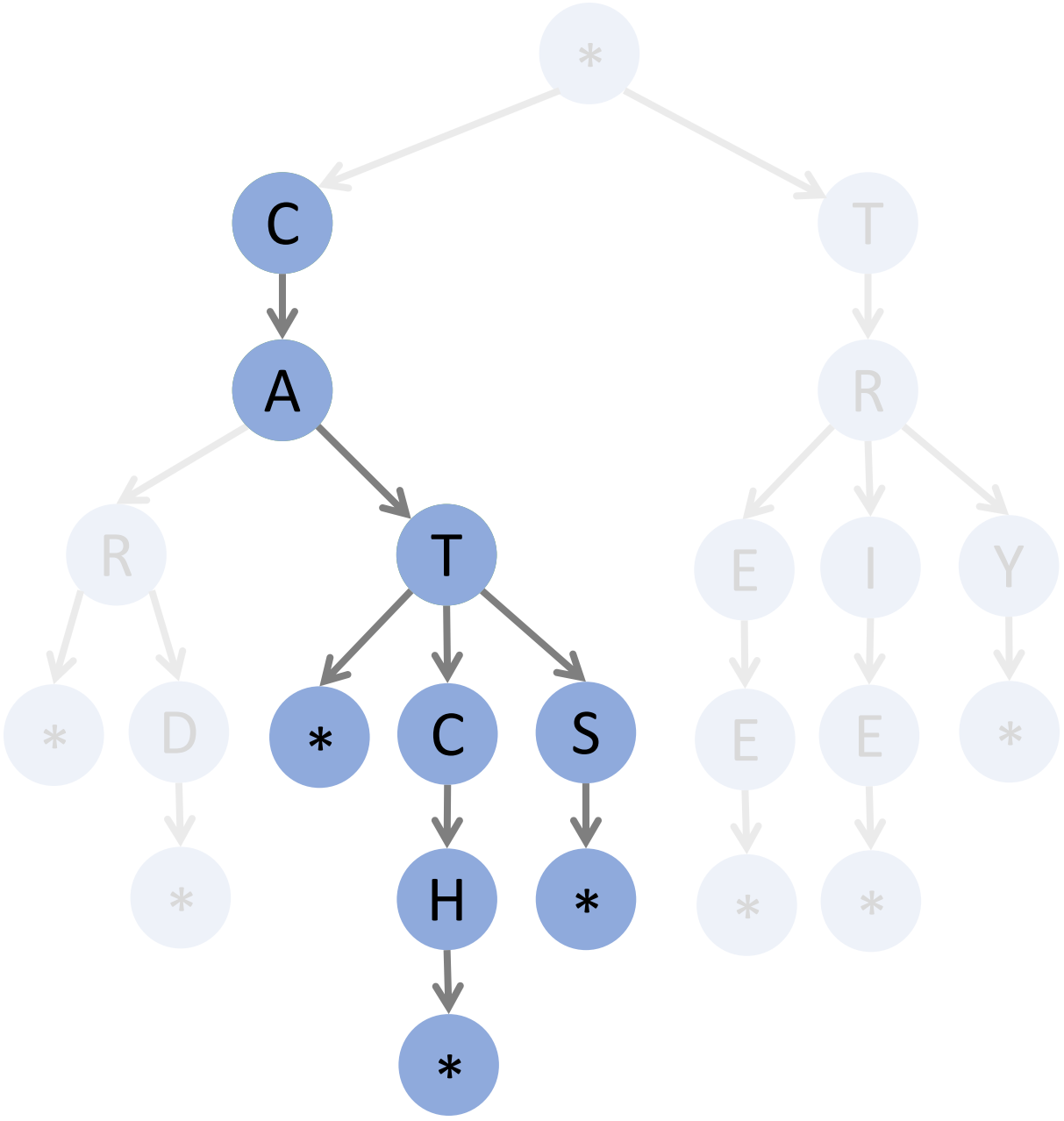
Task 2: Prefix Matching

Prefix Match: Example 1



Query: “ C A T ”

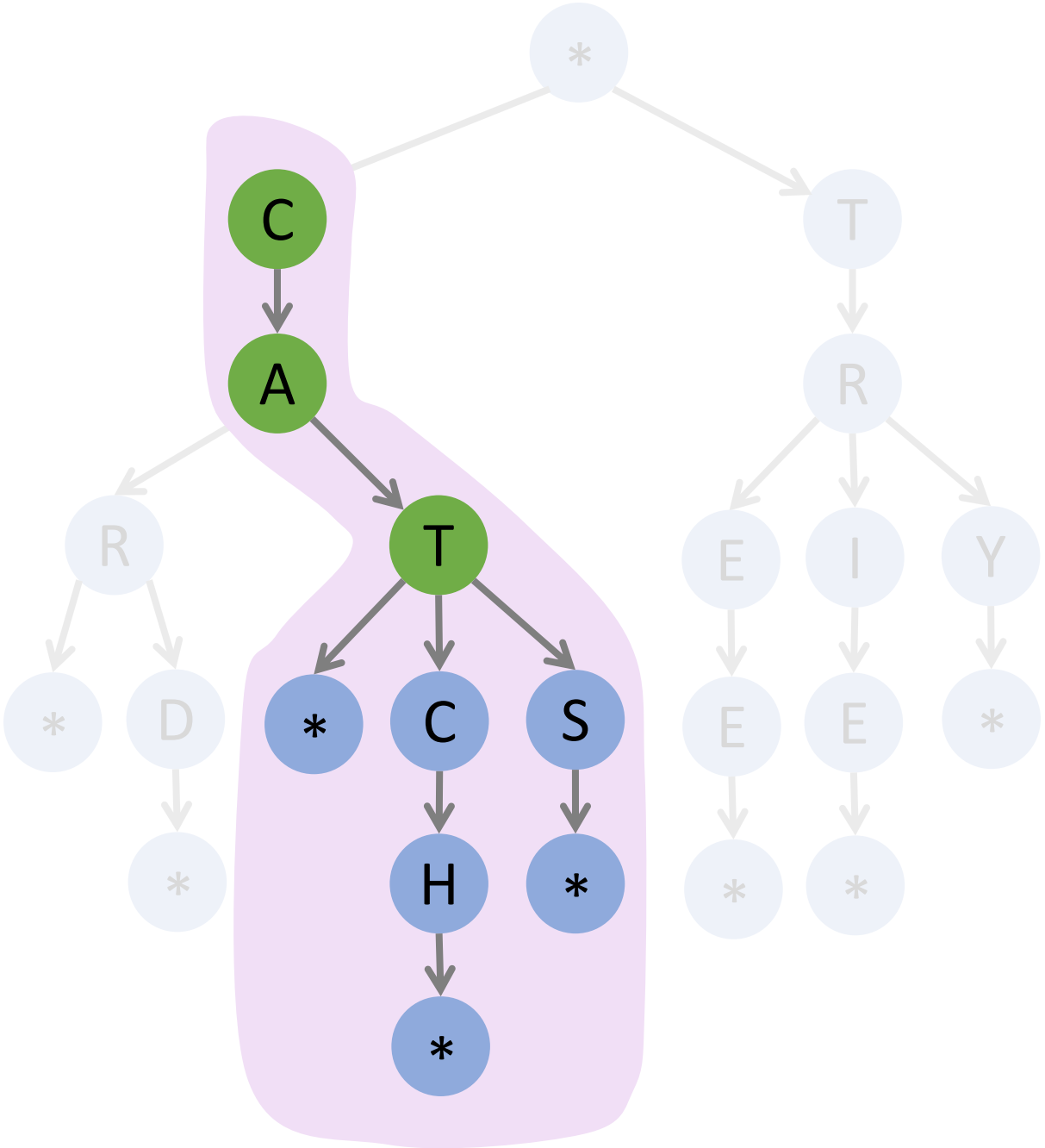
Prefix Match: Example 1



Query: “ C A T ”

- Search “CAT”.
- Found! Return the subtrees.

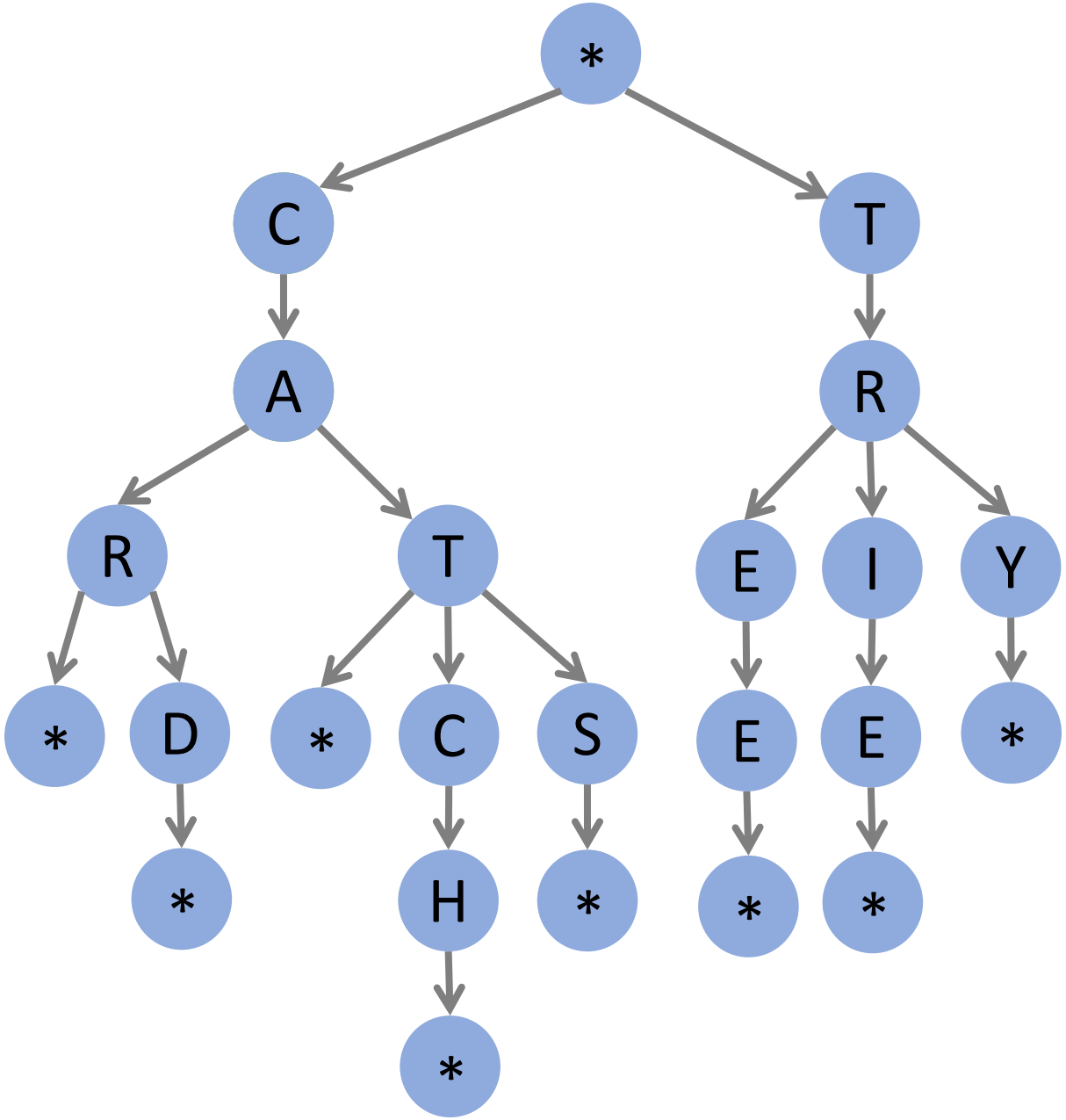
Prefix Match: Example 1



Query: “ C A T ”

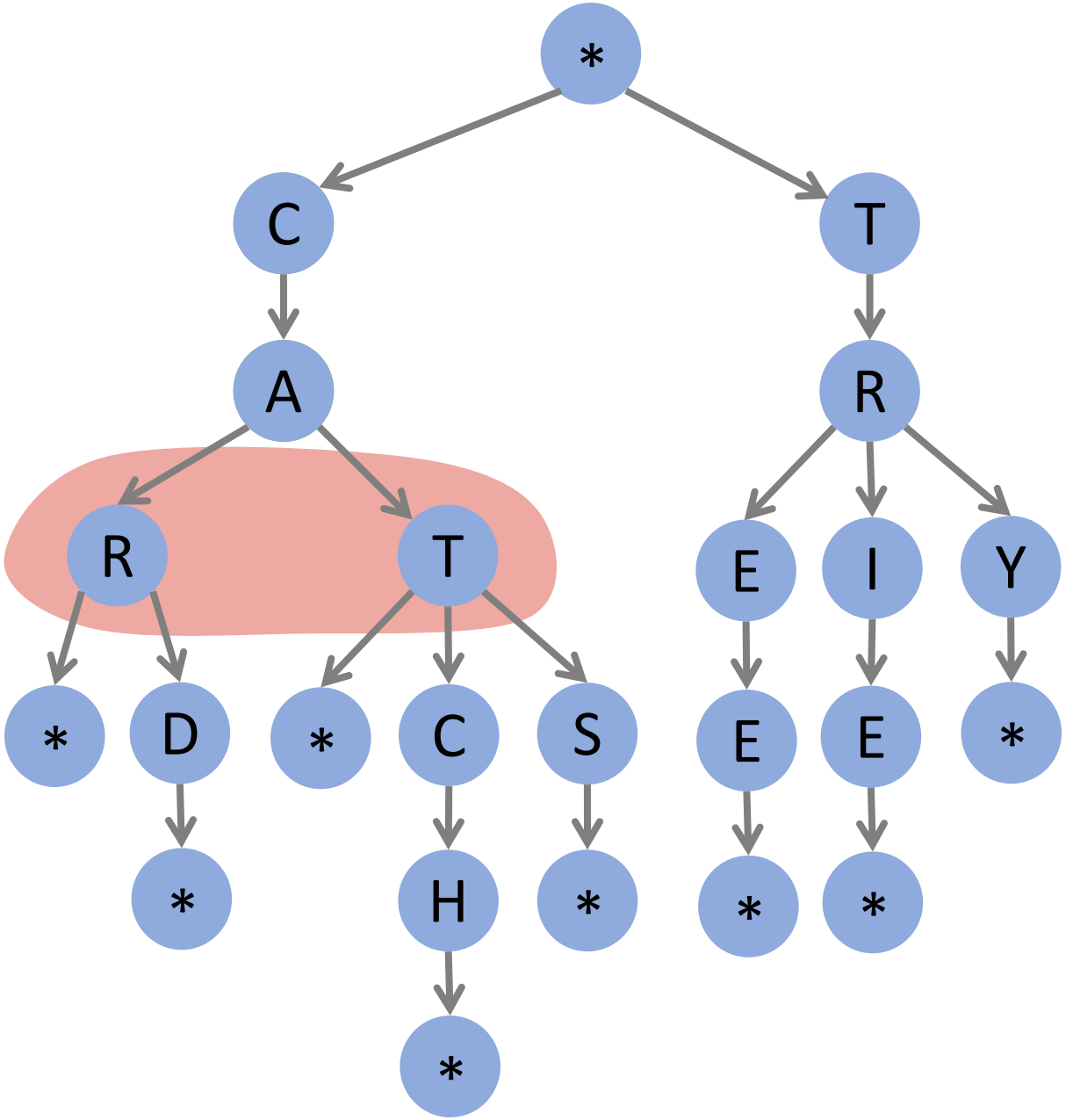
- Search “CAT”.
- Found! Return the subtrees.
- Words that start with “CAT”:
  - “CAT”,
  - “CATCH”,
  - “CATS”.

Prefix Match: Example 2



Query: “ C A B ”

Prefix Match: Example 2

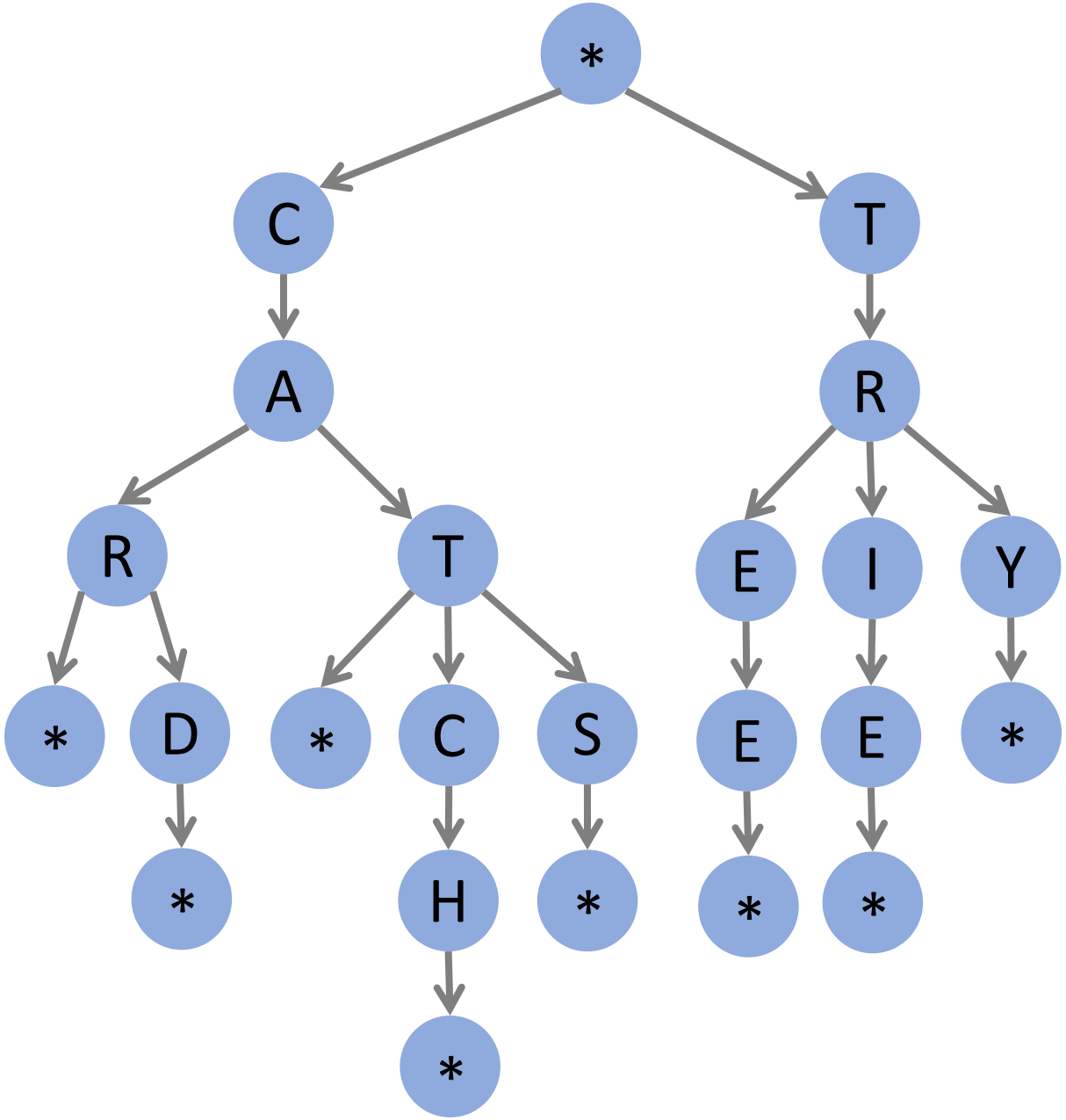


Query: “ C A B ”

- Search “CAB”.
- Not found!
- Return NULL.

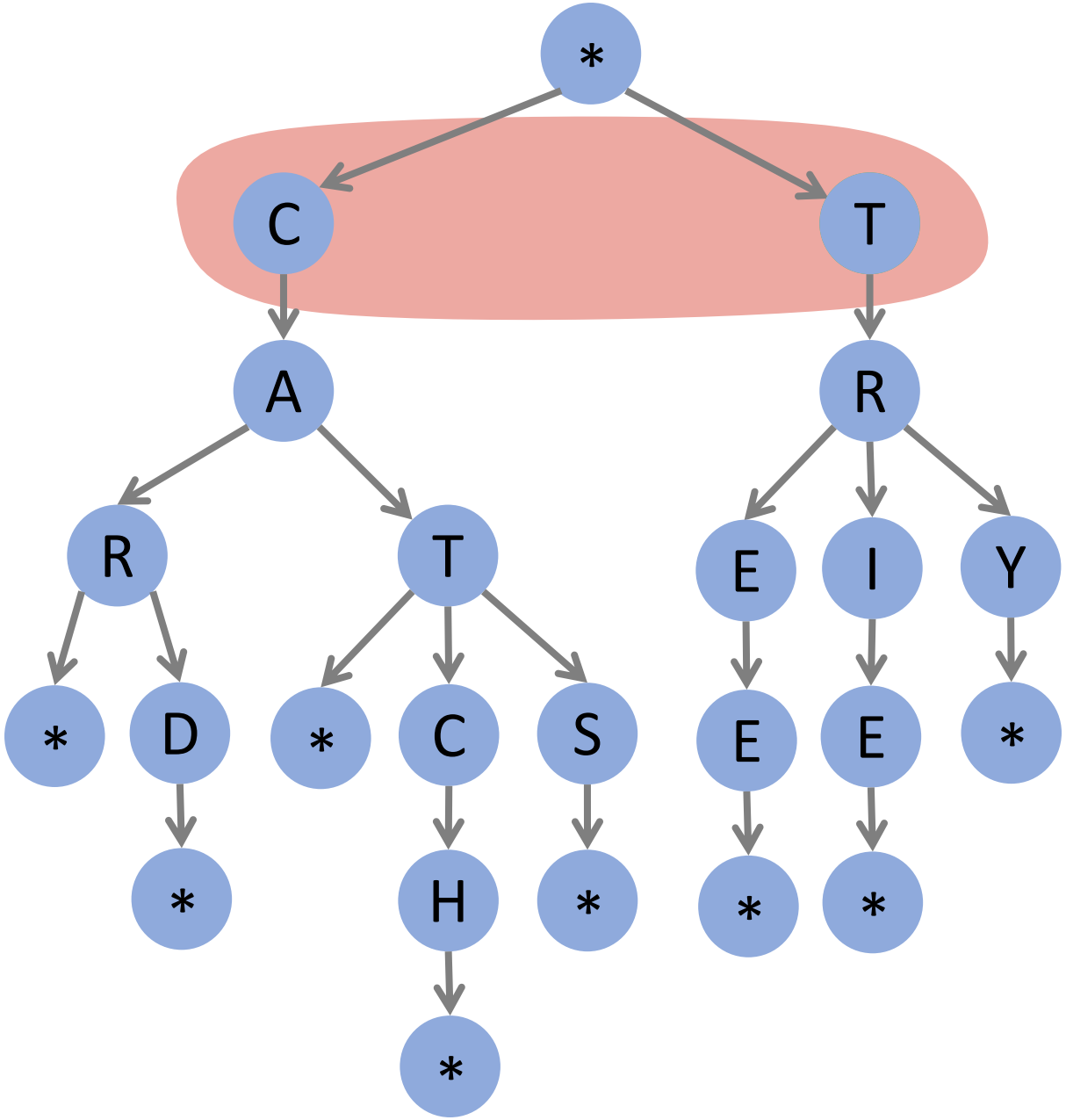
Task 3: Insertion

Insert: Example 1



Insert word “ T E A ”

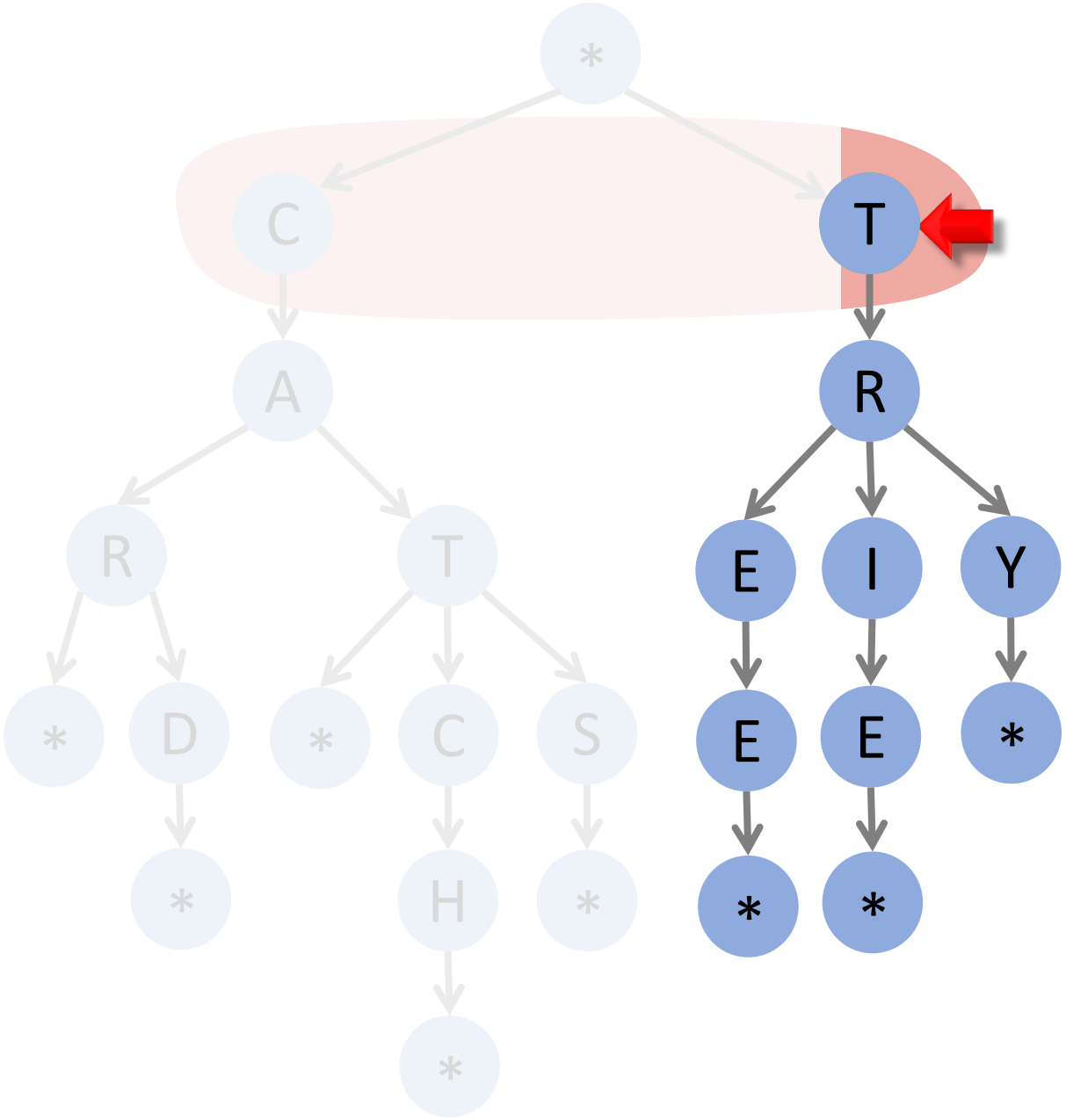
Insert: Example 1



Insert word “ T E A ”

search

Insert: Example 1

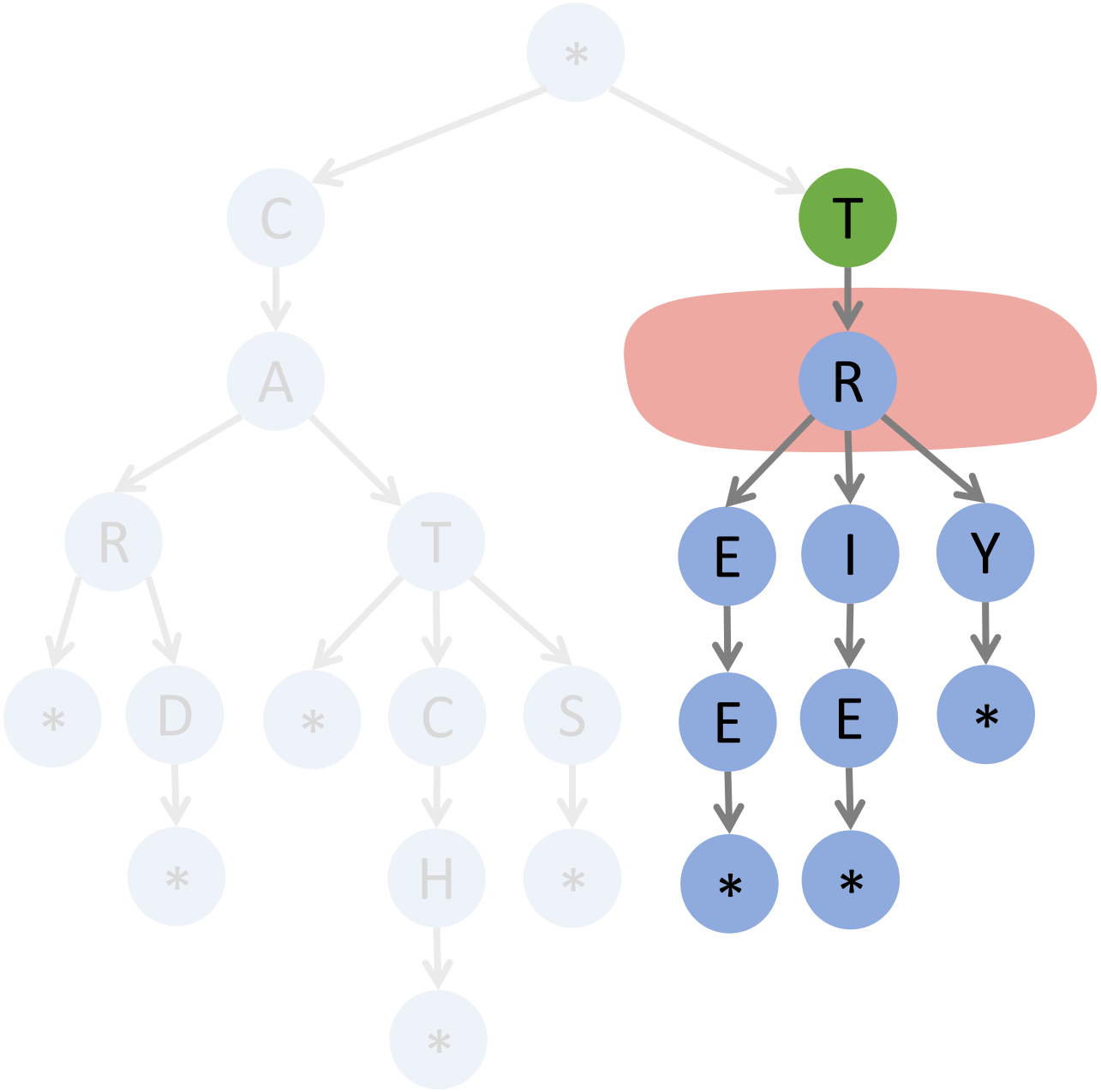


Insert word " T E A "

search

found!

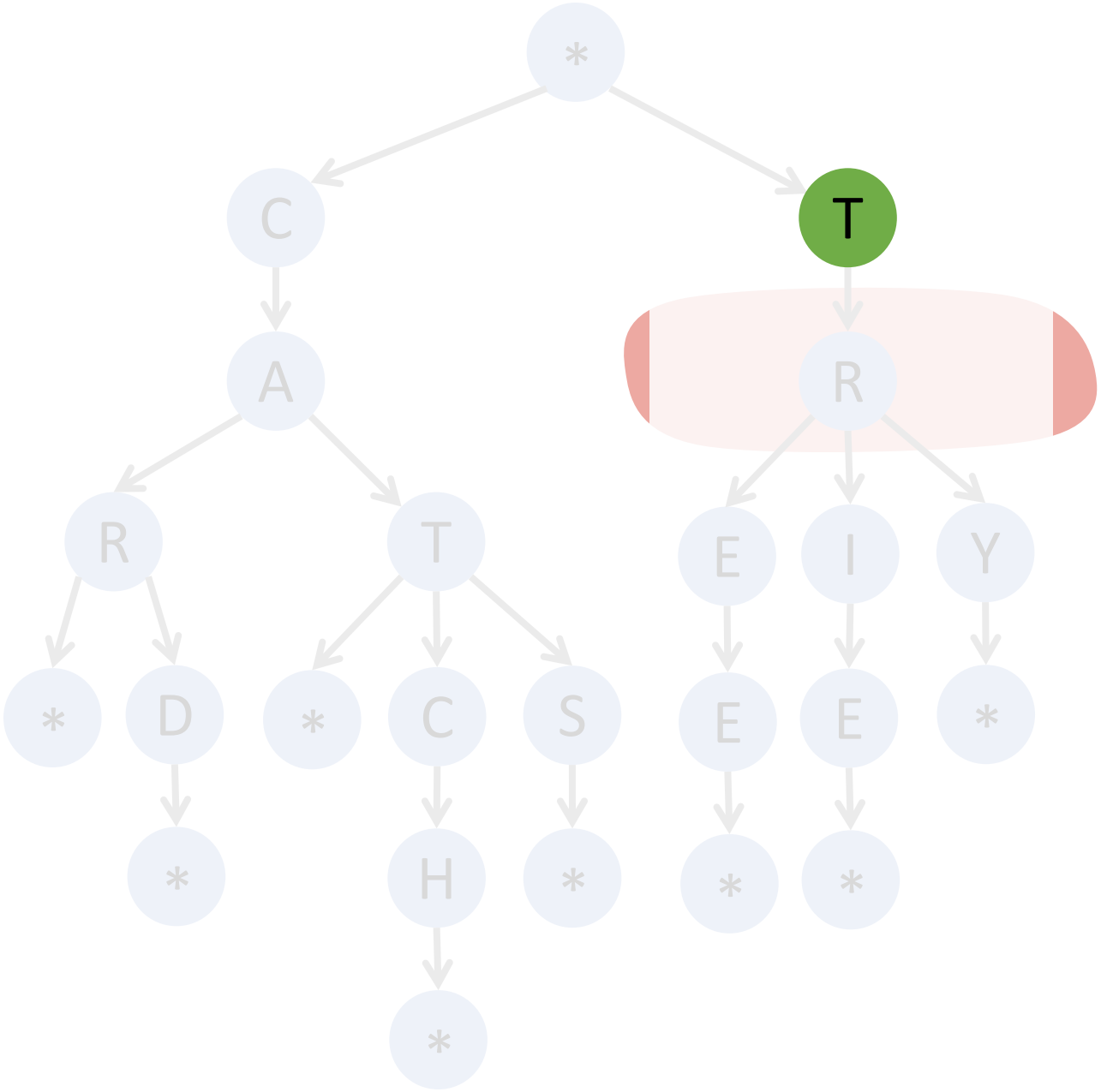
Insert: Example 1



Insert word “ T E A ”

search

Insert: Example 1

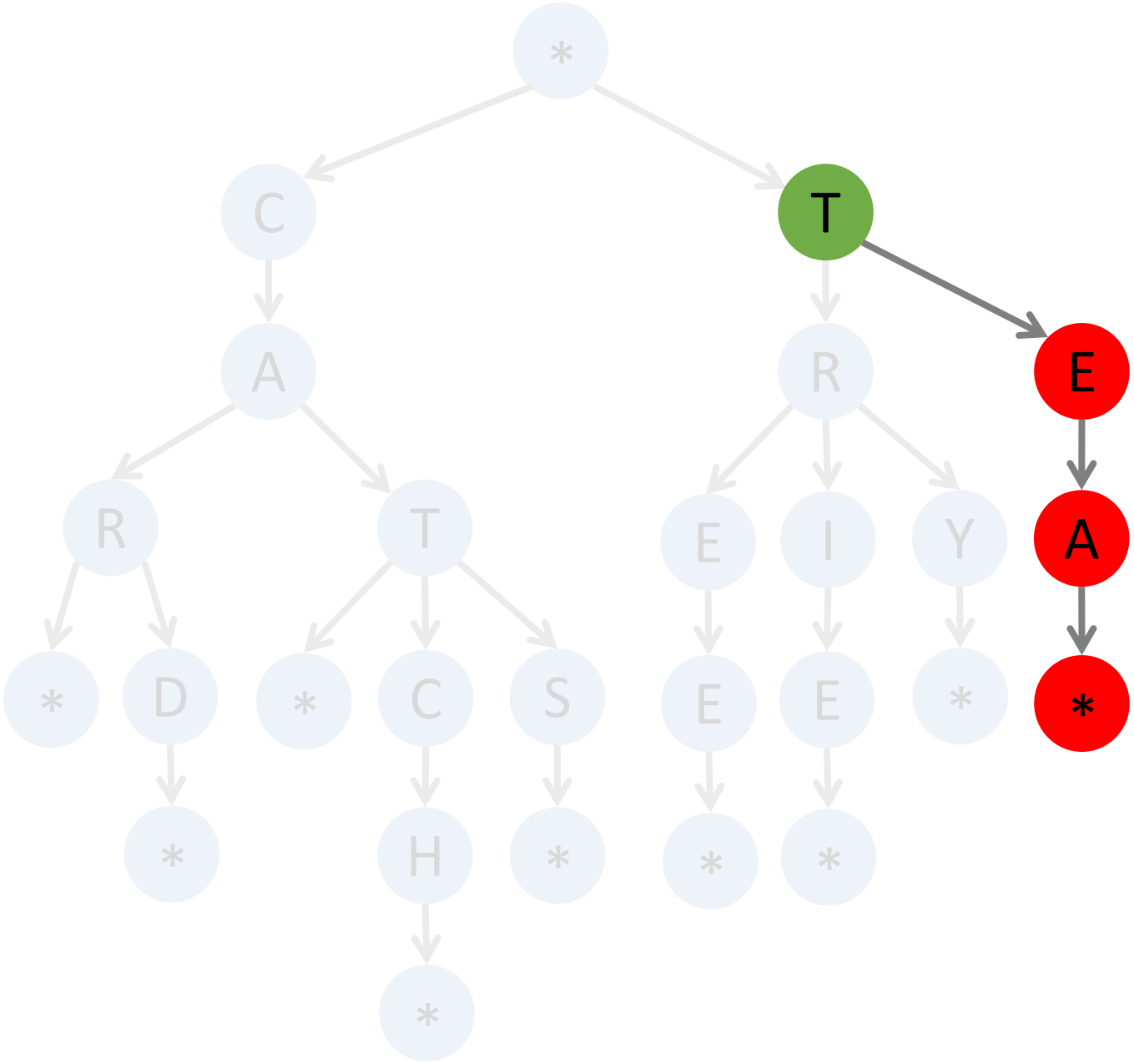


Insert word “ T E A ”

search

not found!

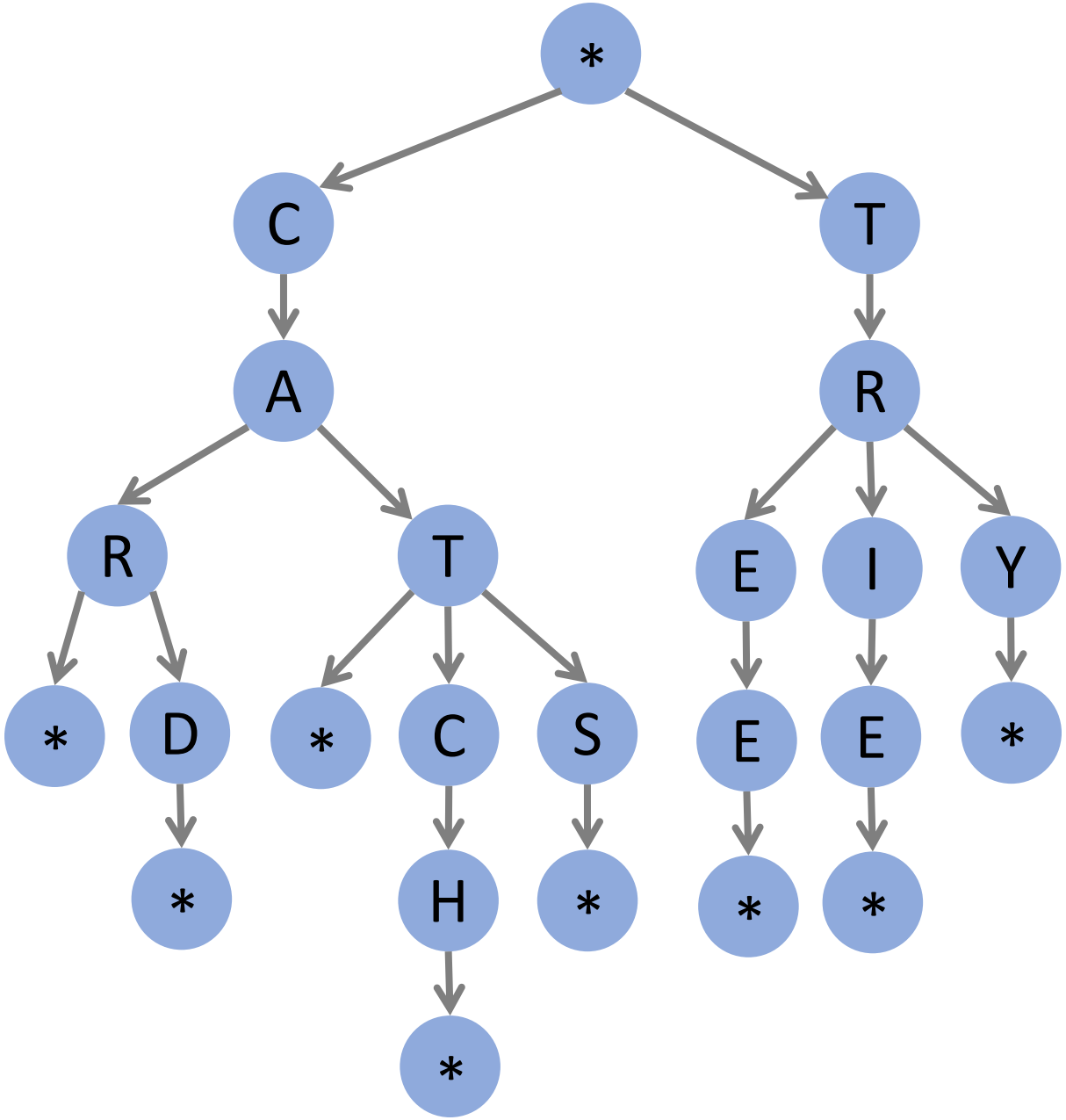
Insert: Example 1



Insert word “ T E A ”

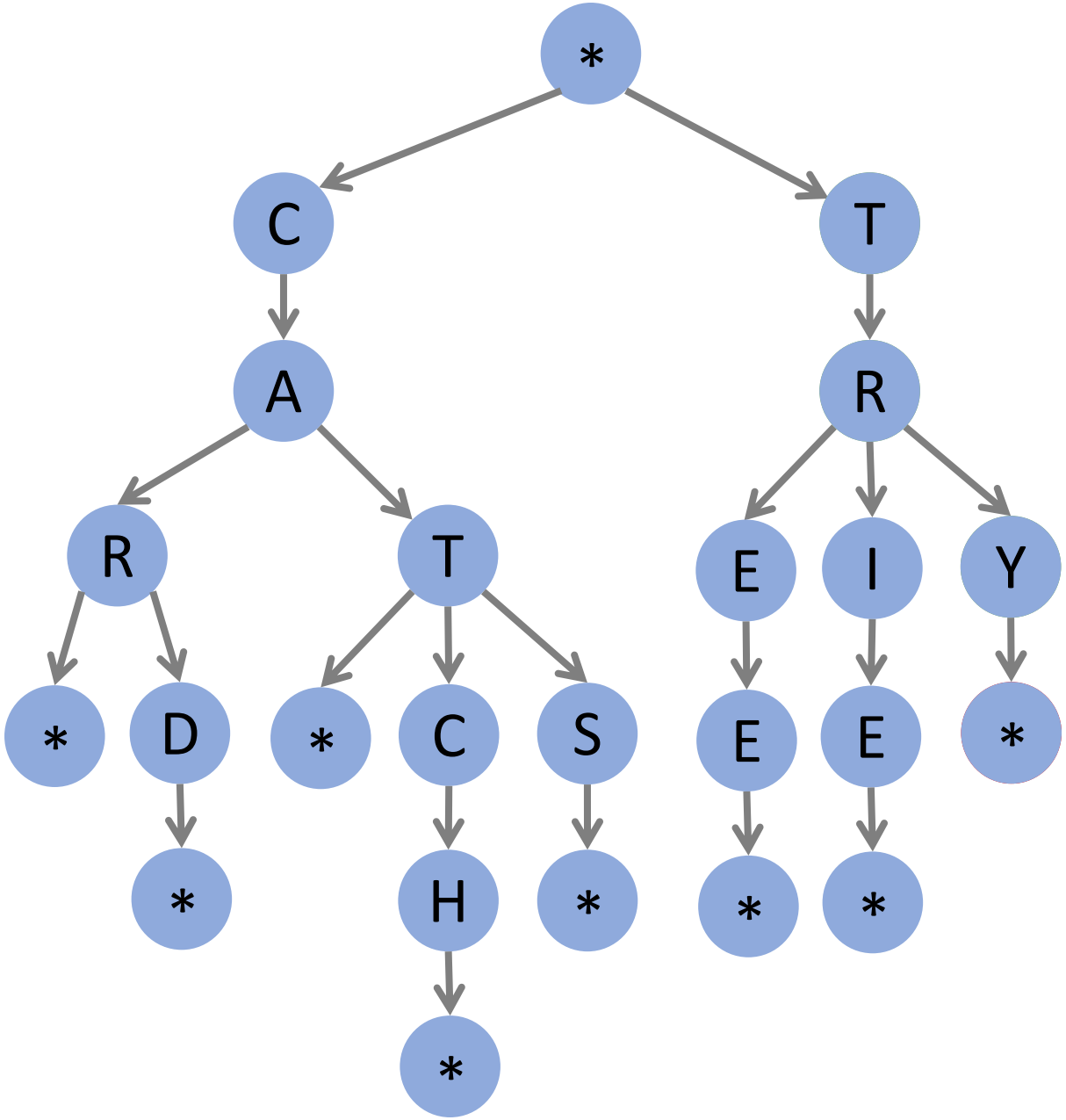
insert

Insert: Example 2



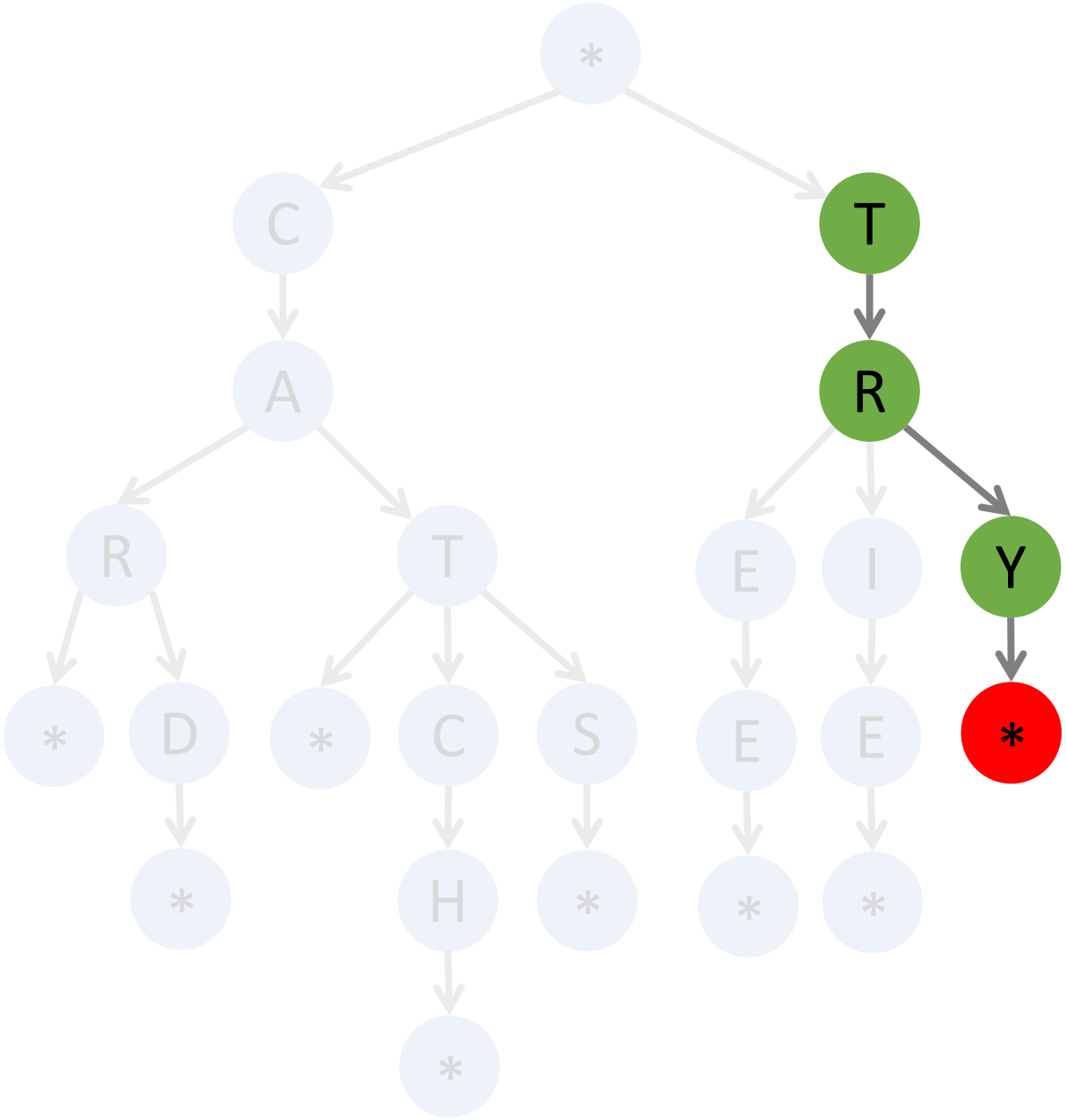
Insert word “ T R Y ”

Insert: Example 2



Insert word “ T R Y ”

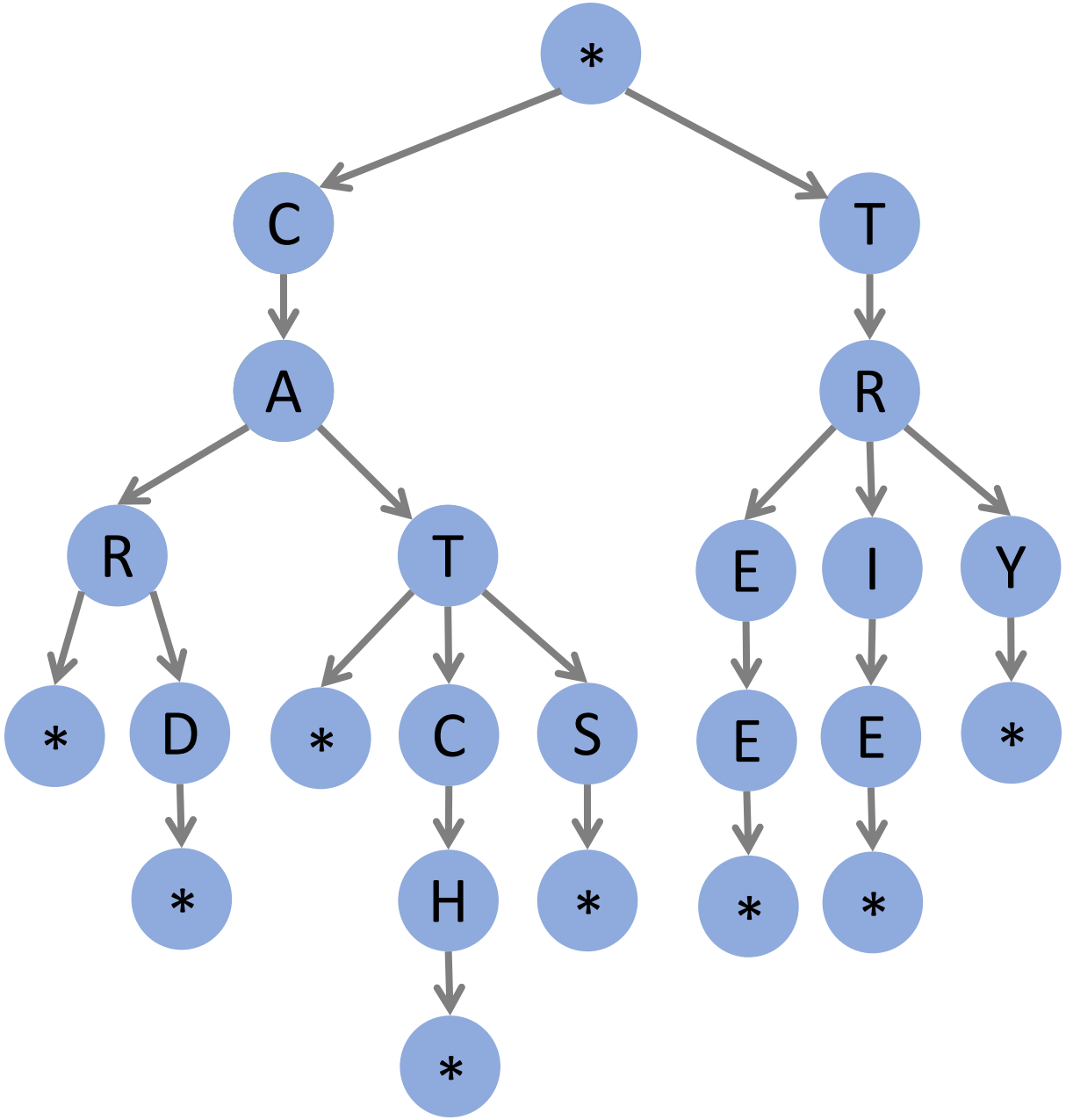
Insert: Example 2



Insert word “ T R Y ”

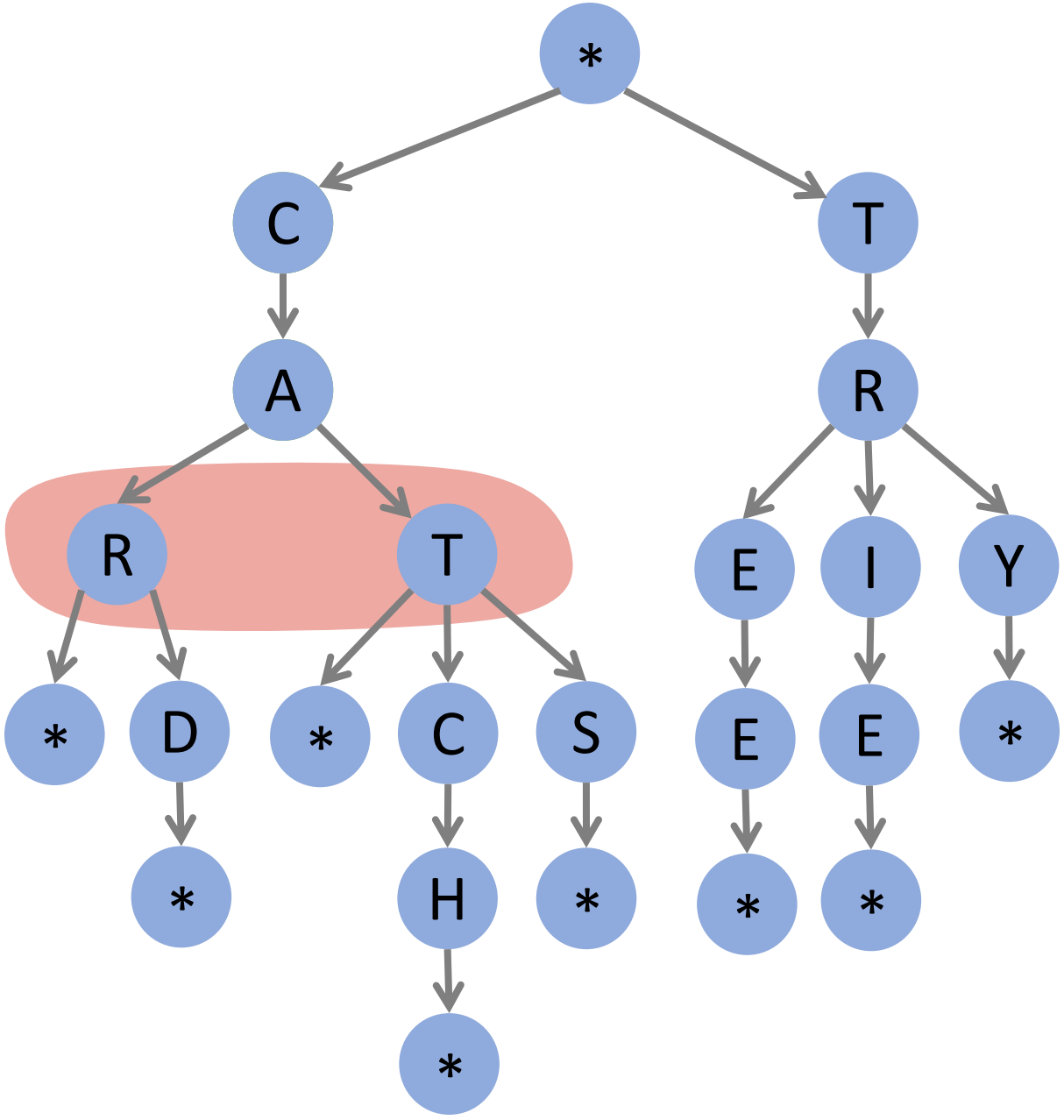
- “TRY” is already in the dictionary.
- No insertion is needed.

Insert: Example 3



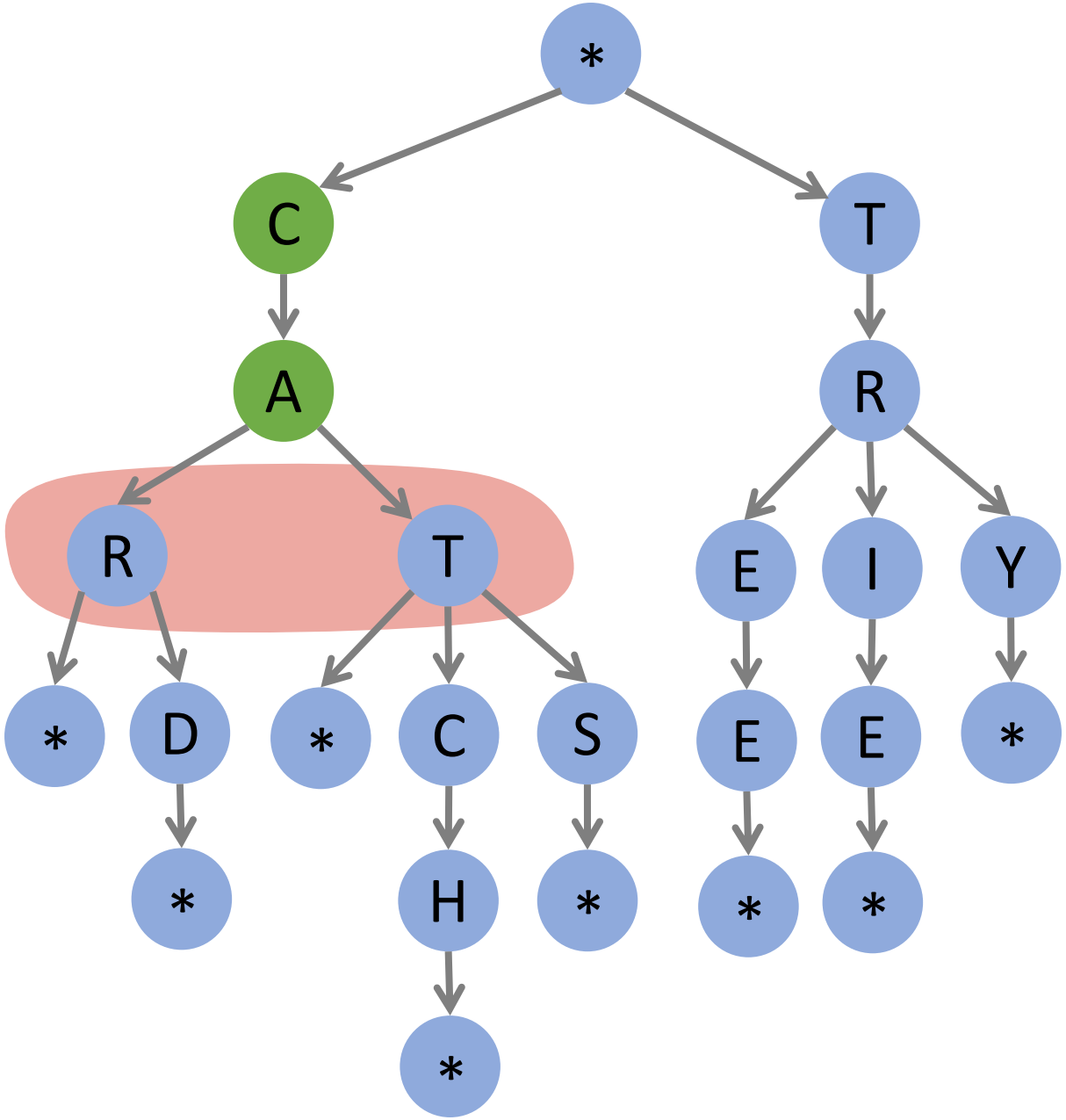
Insert word “ C A ”

Insert: Example 3



Insert word “ C A ”

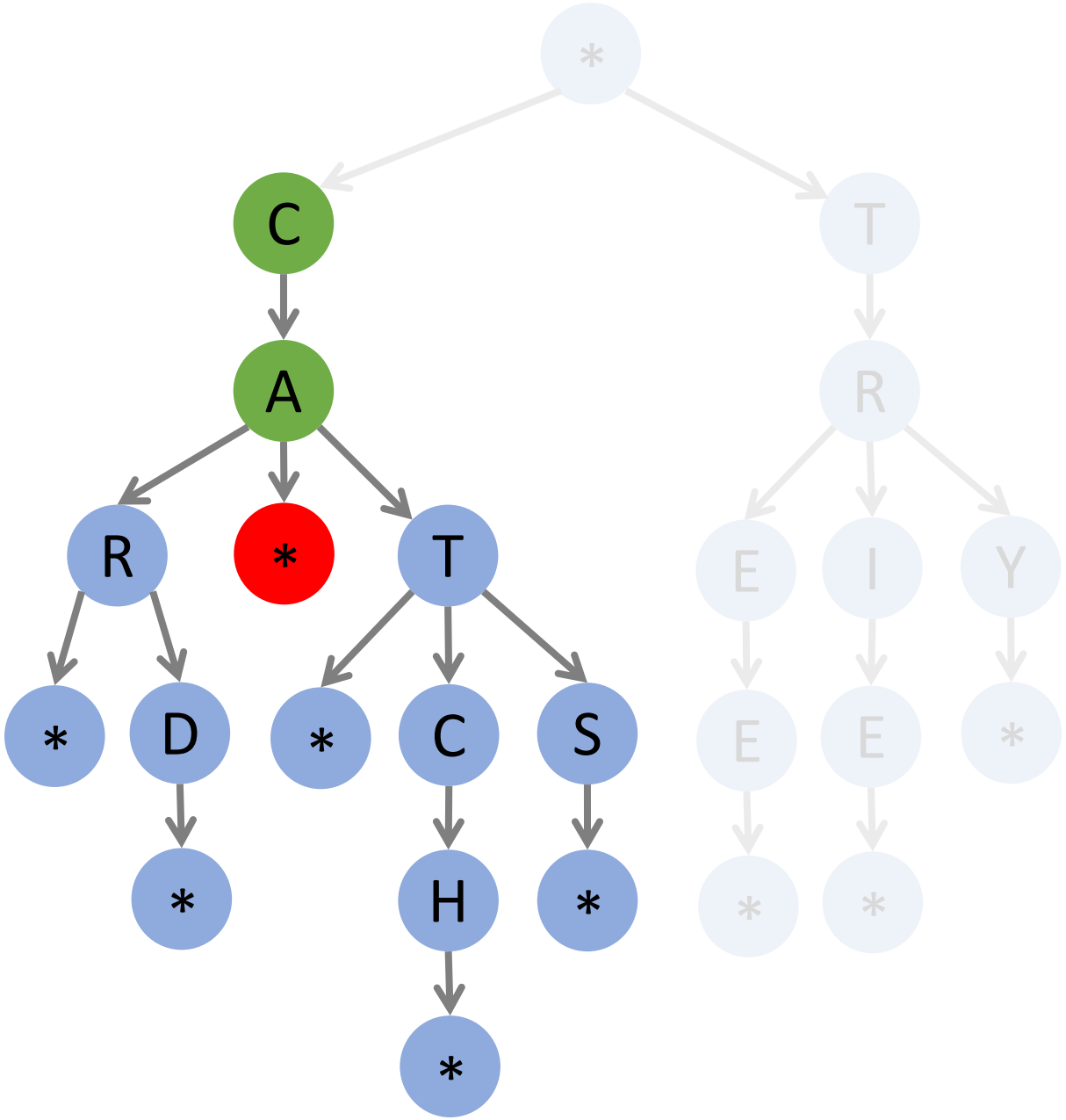
Insert: Example 3



Insert word “ C A ”

- ‘A’ is not the end of word. (‘∗’ is not among its children.)

Insert: Example 3



Insert word “ C A ”

- ‘A’ is not the end of word. (‘∗’ is not among its children.)
- Mark ‘A’ as the end of word.

Implementation

Vertex Representation

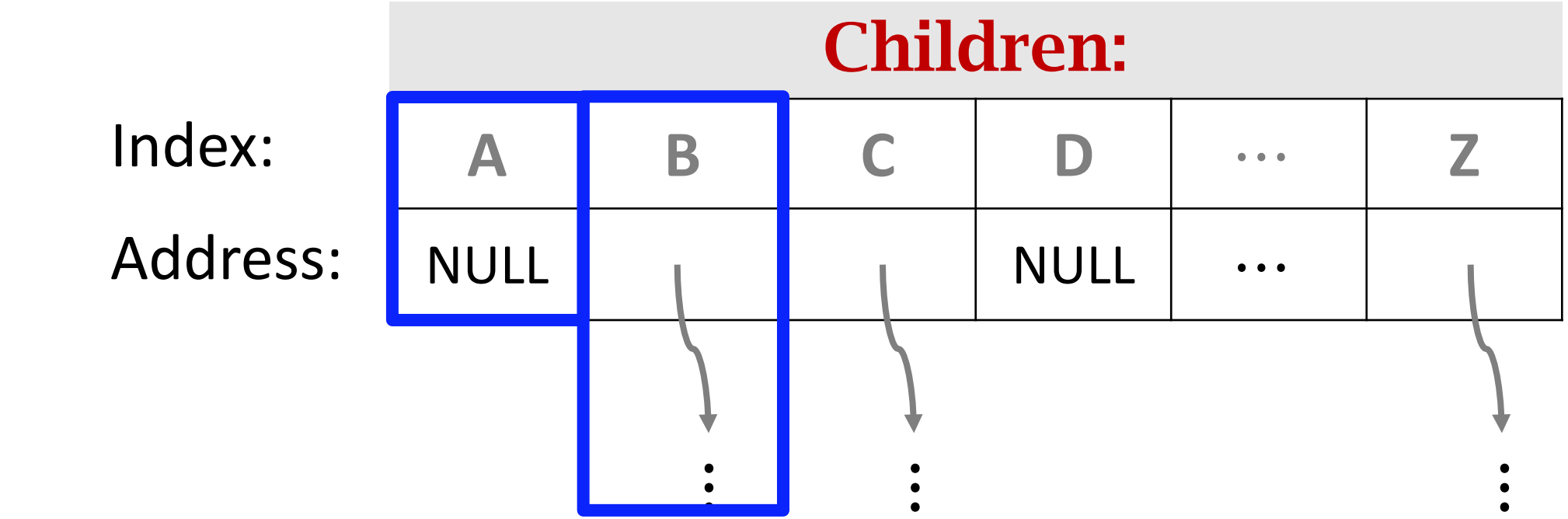
```
struct Vertex {
    bool isEndOfWord;
    struct Vertex *children[ALPHABET_SIZE];
};
```

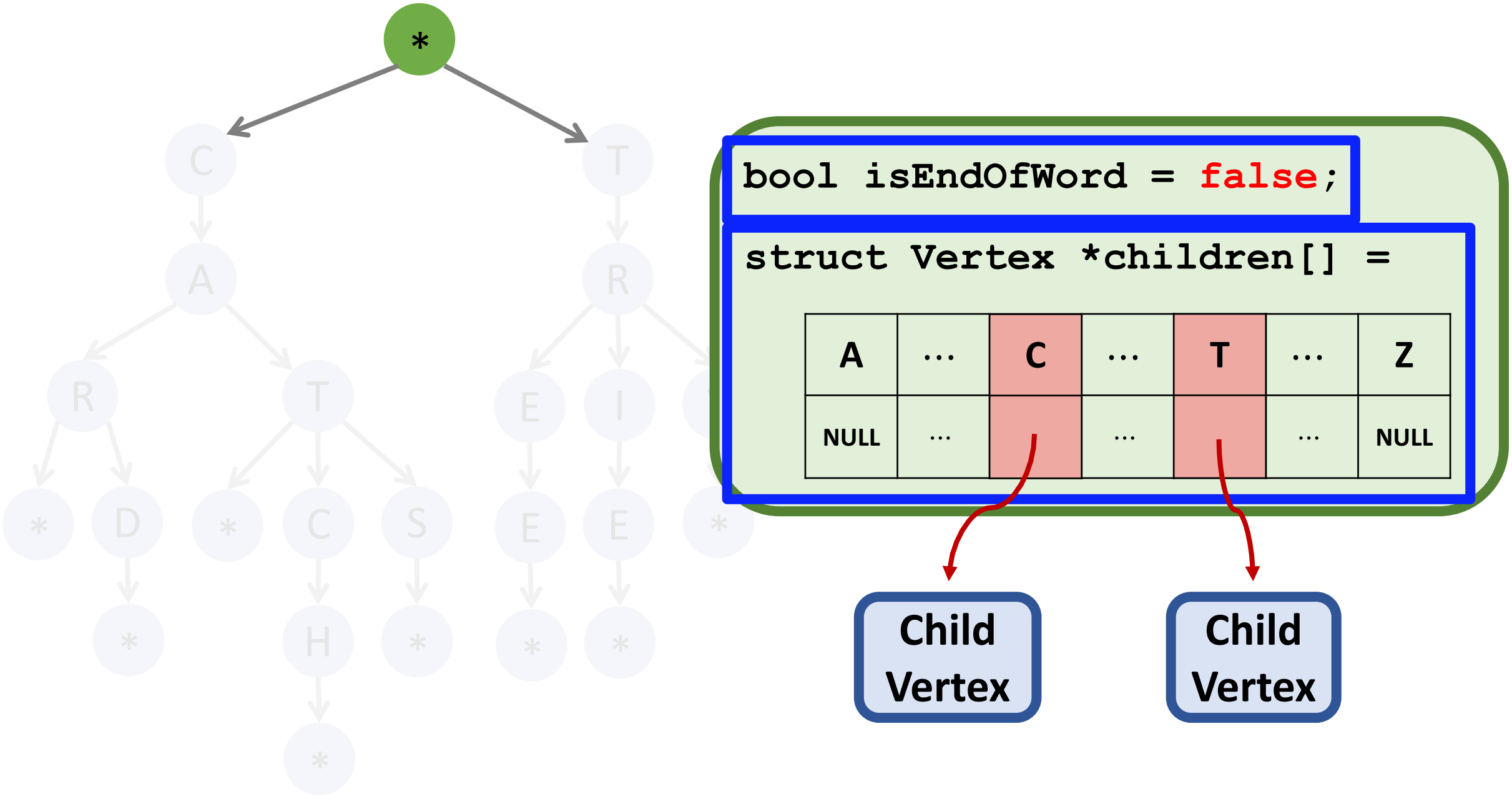
Vertex Representation

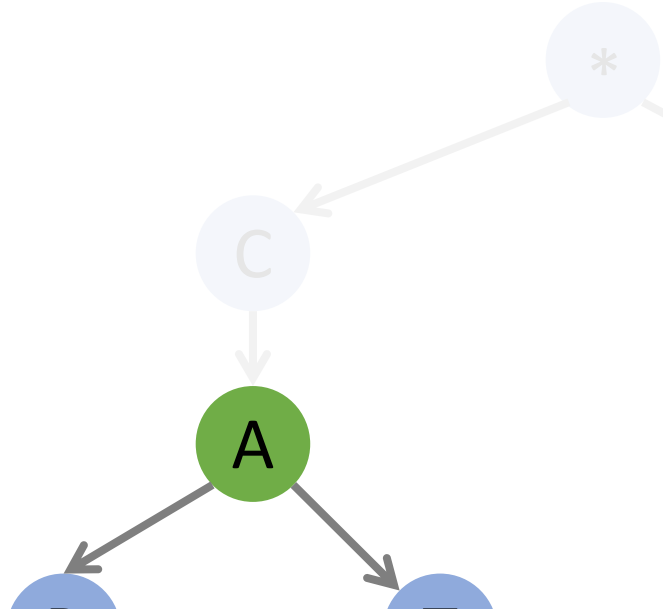
```
struct Vertex {
    bool isEndOfWord;
    struct Vertex *children[ALPHABET_SIZE];
};
```

Vertex Representation

```
struct Vertex {
    bool isEndOfWord;
    struct Vertex *children[ALPHABET_SIZE];
};
```



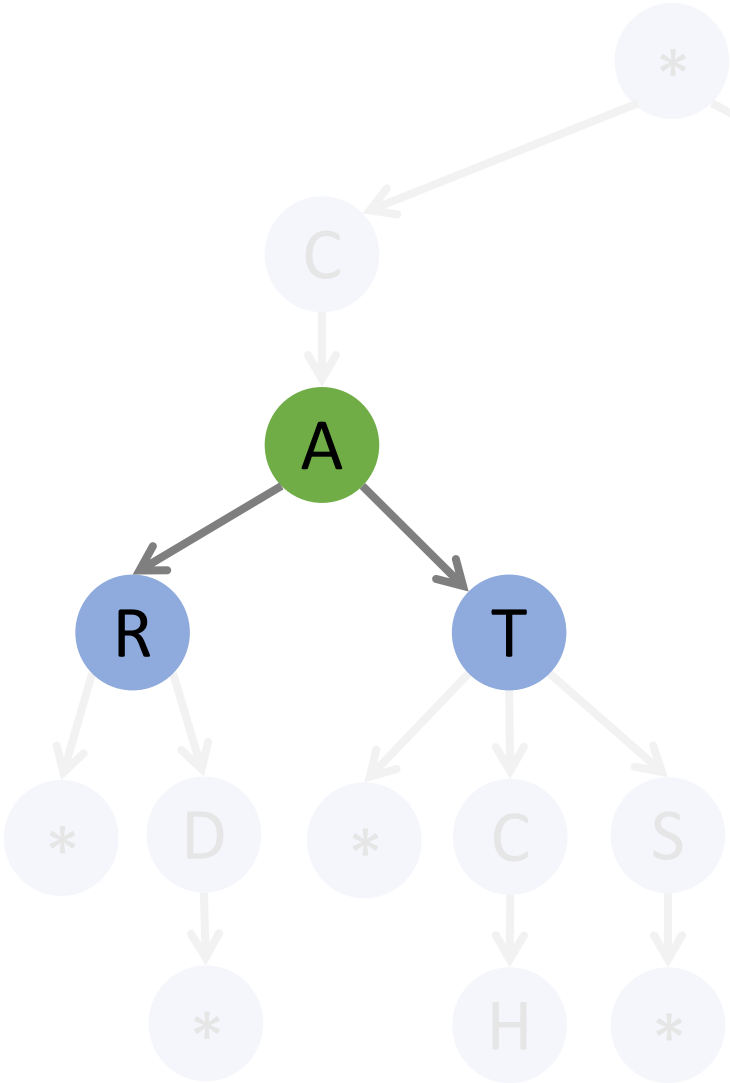

*
C
T
A
R
R
T
E
I
*
D
*
C
S
E
E
*
*
H
*
*
*
*
*
bool isEndOfWord = false;
struct Vertex *children[] =
A
…
C
…
T
…
Z
NULL
…
…
…
NULL
Child Vertex
Child Vertex




```
bool isEndOfWord = false;

struct Vertex *children[] =
```

A	…	R	S	T	…	Z
NULL	…		NULL		…	NULL



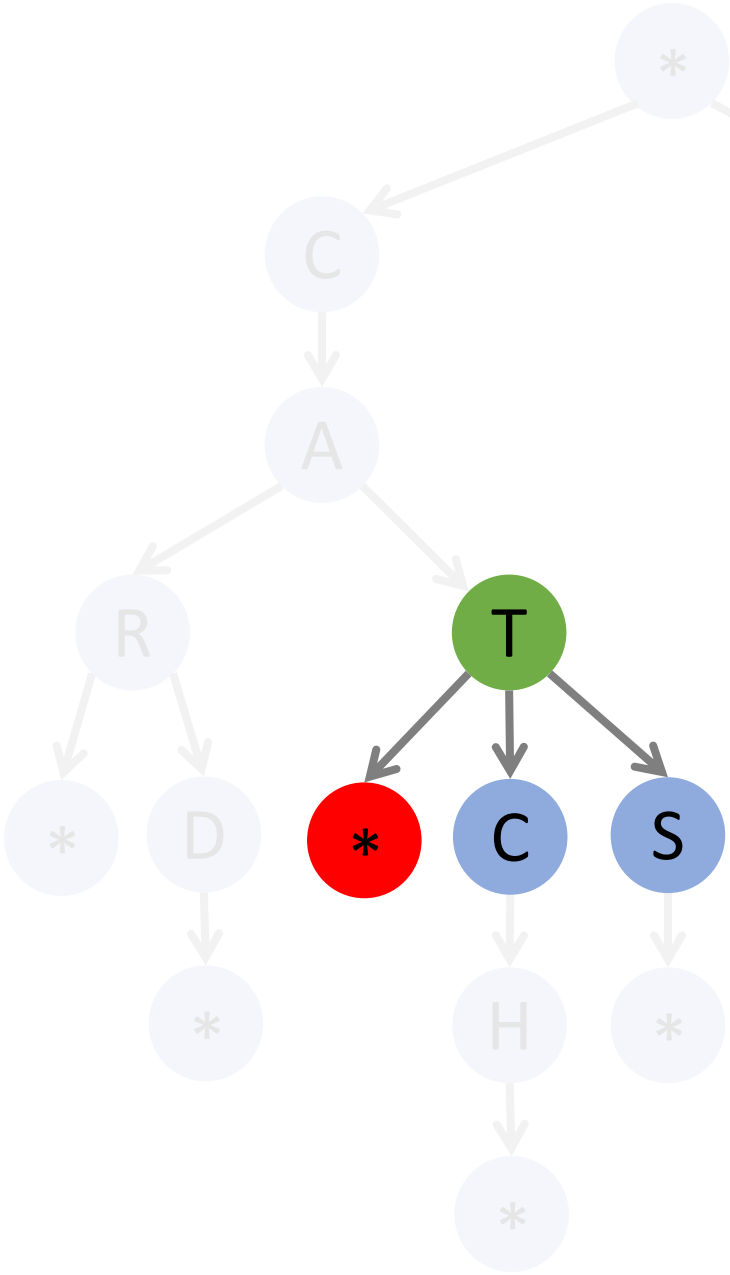

```
bool isEndOfWord = false;

struct Vertex *children[] =
```

A	…	R	S	T	…	Z
NULL	…		NULL		…	NULL

Child Vertex

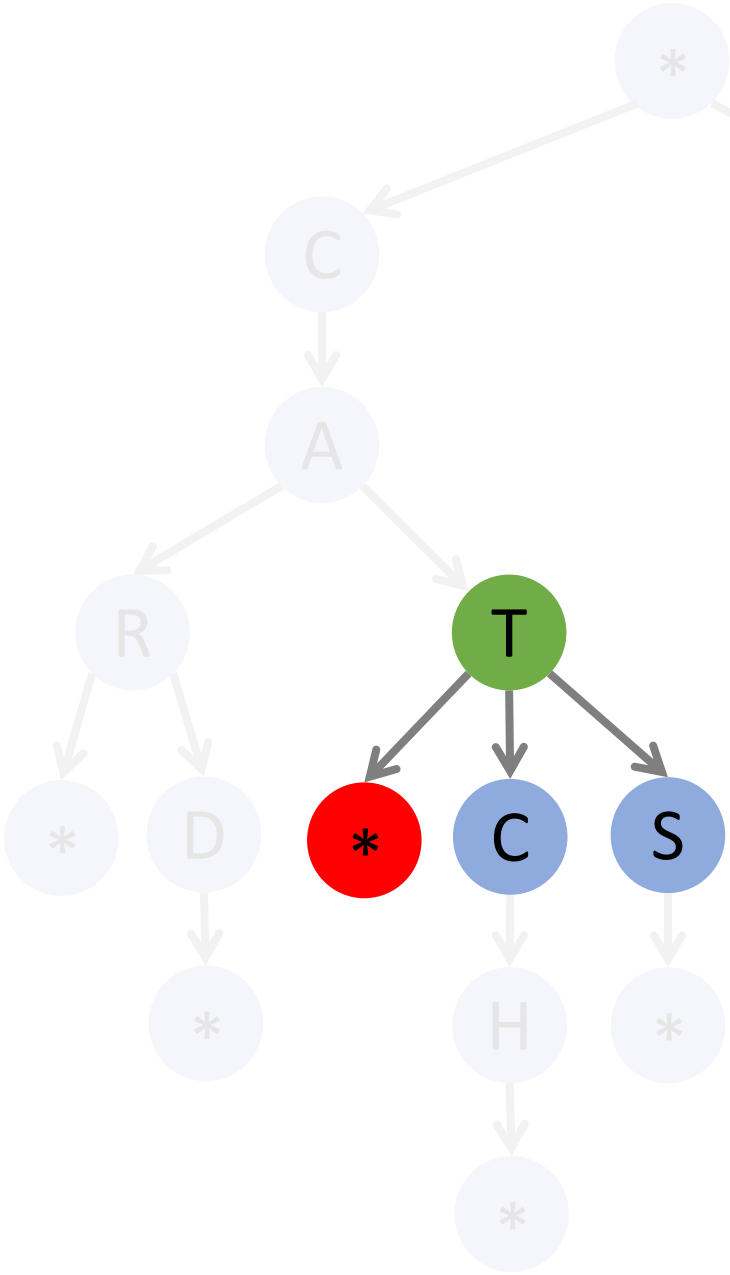
Child Vertex




```
bool isEndOfWord = true;

struct Vertex *children[] =
```

A	…	C	…	S	…	Z
NULL	…		…		…	NULL



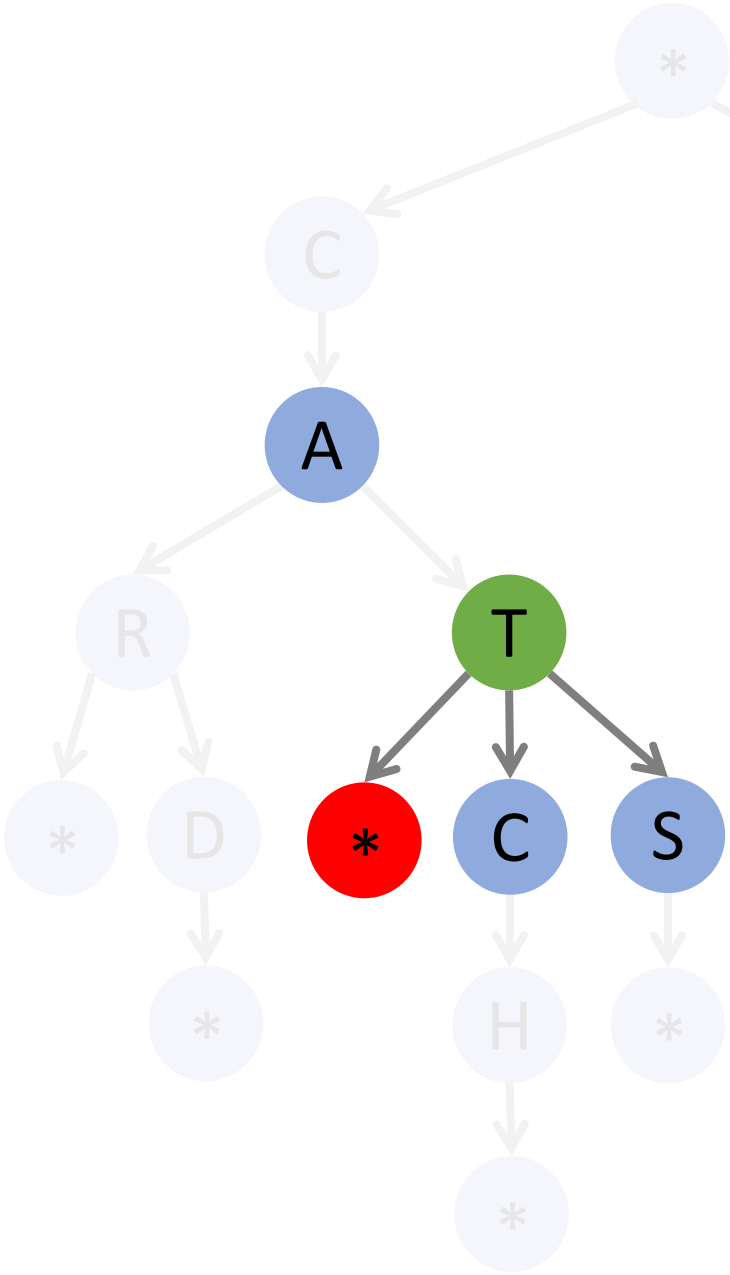

```
bool isEndOfWord = true;

struct Vertex *children[] =
```

A	…	C	…	S	…	Z
NULL	…		…		…	NULL

Child Vertex

Child Vertex




```
bool isEndOfWord = true;
struct Vertex *children[] =
```

A	…	C	…	S	…	Z
NULL	…		…		…	NULL

Child Vertex

Child Vertex

Questions

Draw the trie

Draw the trie that results when the following words are inserted:

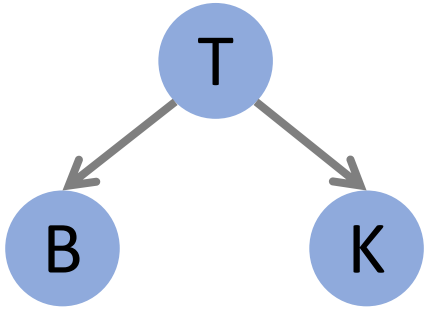
- **Words:** “SHIN”, “SAKE”, “SHAPE”, “SPIN”, “SPEED”, “SAM”, “SPEAK”, “SHA”, “SPIDER”, “SALE”, “SPY”.

Thank You!

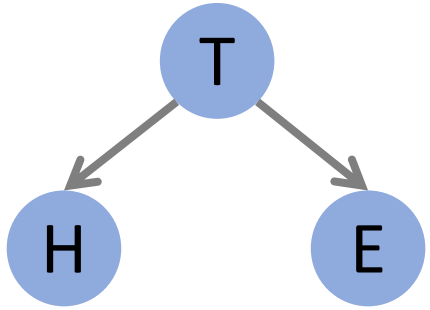
Draw the trie

Draw the trie that results when the following words are inserted:

- **Words:** “SHIN”, “SAKE”, “SHAPE”, “SPIN”, “SPEED”, “SAM”, “SPEAK”, “SHA”, “SPIDER”, “SALE”, “SPY”.
- **Requirement:** Children of a node must have the lexicographical order.



Correct Order



Incorrect Order