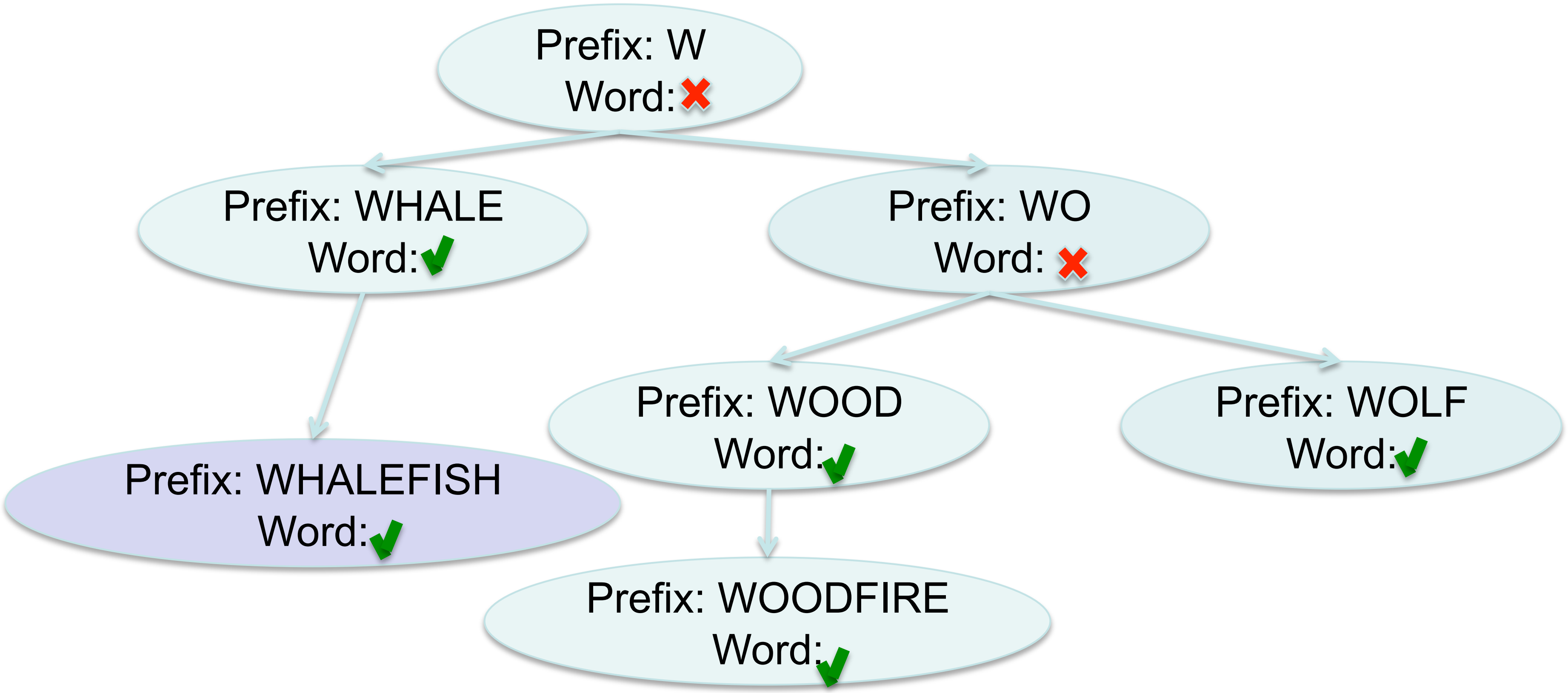
C & OpenMP Laboratory

Daniel Lorenz



Organization

You have three weeks to complete the tasks

- Submit your solution to Moodle until December 13

The labortory is graded

You as a group have to present the solution to a tutor

- Presentations of solutions within 3 weeks after submission
- We will assign groups to tutors
- The whole group must be present
- Everybody must be able to explain every part of the code

Base idea

Development of a tree-based dictionary

The dictionary is constructed from the words in a text file

The program reads a second text file and counts all words that do not appear in the dictionary

- A word that appears twice is counted twice

General remarks

We provide header files for download

- You need to implement the specified functions and data types
- You must not modify the header files
  - In a test you can not change the questions!

You need to provide a Makefile to build for every program

- Even if tasks build on top of each other, you need to provide all intermediate steps

You must free all dynamically allocated memory

You can download larger text files e.g., from

www.gutenberg.org

Task overview

1. Implementation of a serial version
2. Parallelize the counting of words that are not in the dictionary
3. Parallelize the construction of the dictionary

Task 1

a. Implement a double-linked list

b. Open a text file and store the text in blocks of 16kB in a double-linked list.

c. Implement a text parser that identifies words from the text

d. Implement the tree-based dictionary and a program that constructs a dictionary from the words in a text file

e. Create a program that reads a second text file and count all words that are not contained in the dictionary

Task 2

a. Parallelize the counting of words that are not in the dictionary with an OpenMP for-loop
   - Every iteration shall process a text block
   - The partial results are summed up with a reduction

b. Measure the execution time of the parallel region from task 2a with different numbers of threads

c. Parallelize the counting of words with OpenMP tasks
   - Every task shall process a text block

d. Parallelize the counting of words with OpenMP tasks.
   - Every task shall process one word
   - Let one thread parse the text and create a task per word

Task 3

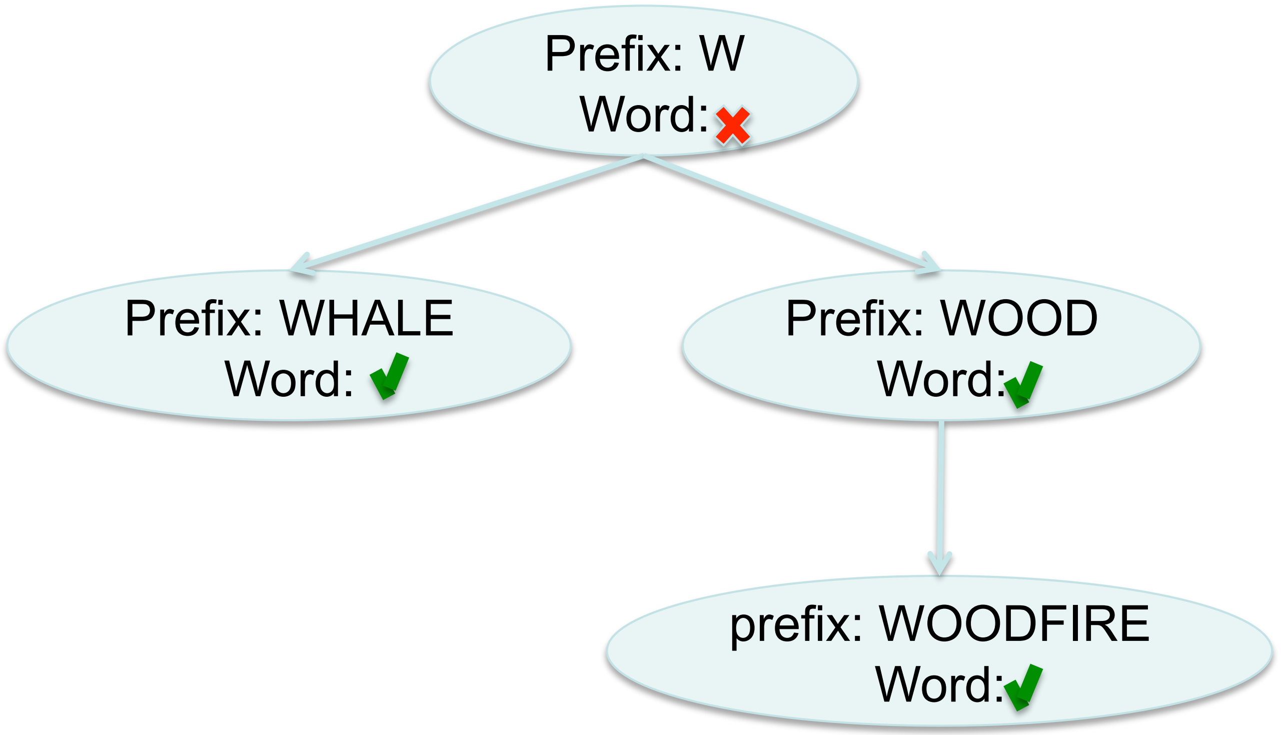
a. Parallelize the construction of the dictionary with an OpenMP for-loop
   - Every iteration shall process a text block
   - Accesses to the commen dictionary are protected with a critical section

b. Instead of working on a common dictionay, every thread shall create a thread-local dictionary
   - Merge the thread-local dictionaries at the end

c. Parallelize the merge of the thread-local dictionaries with a divide & conquer strategy using OpenMP tasks

The dictionary (1)

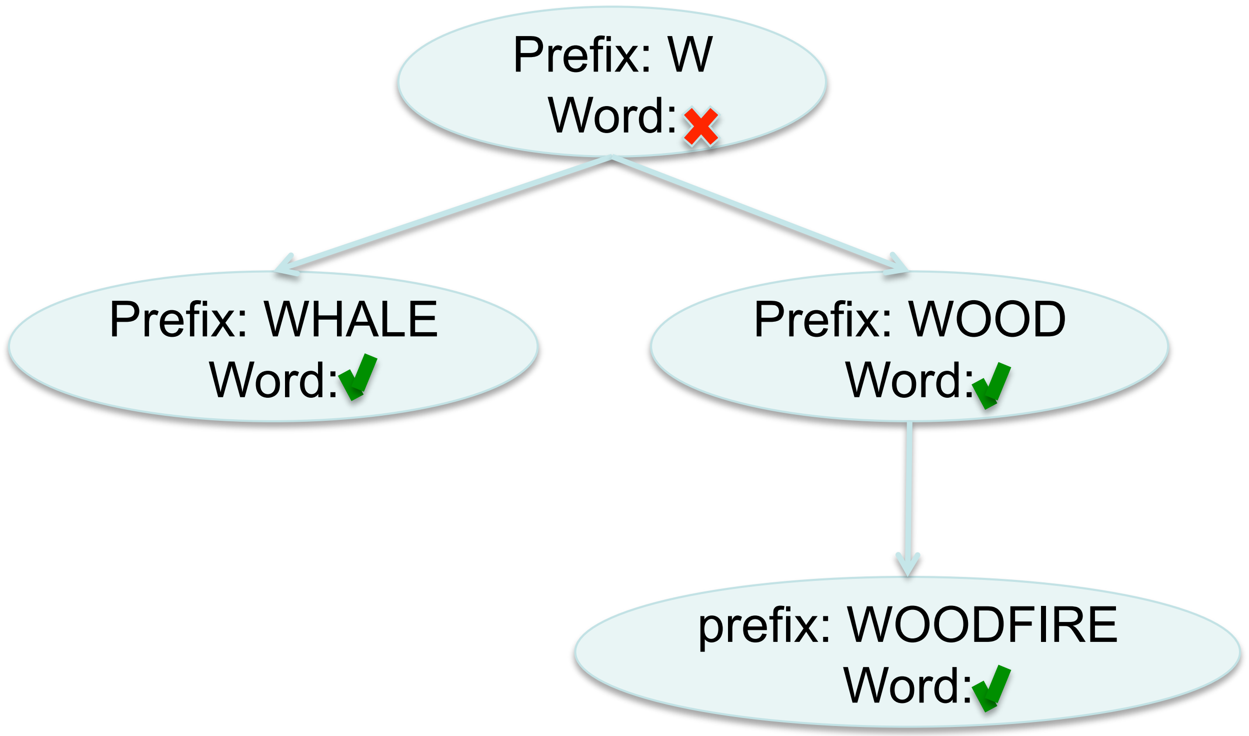
Every node stores a common prefix for all words in its sub-tree



A dictionary with the words WHALE, WOOD and WOODFIRE

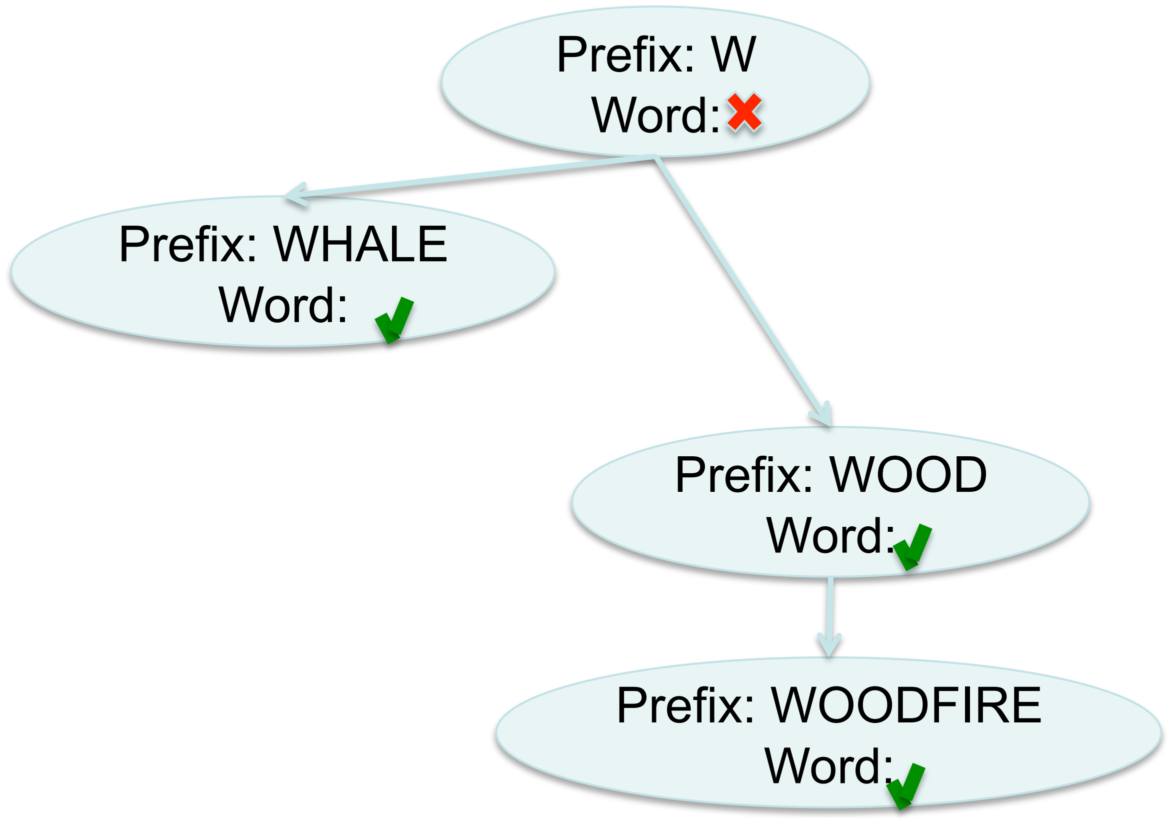
The dictionary (2)

The dictionary must mark nodes that contain a word of the dictionary



WOOD is a word in the dictionary and a prefix for WOODFIRE

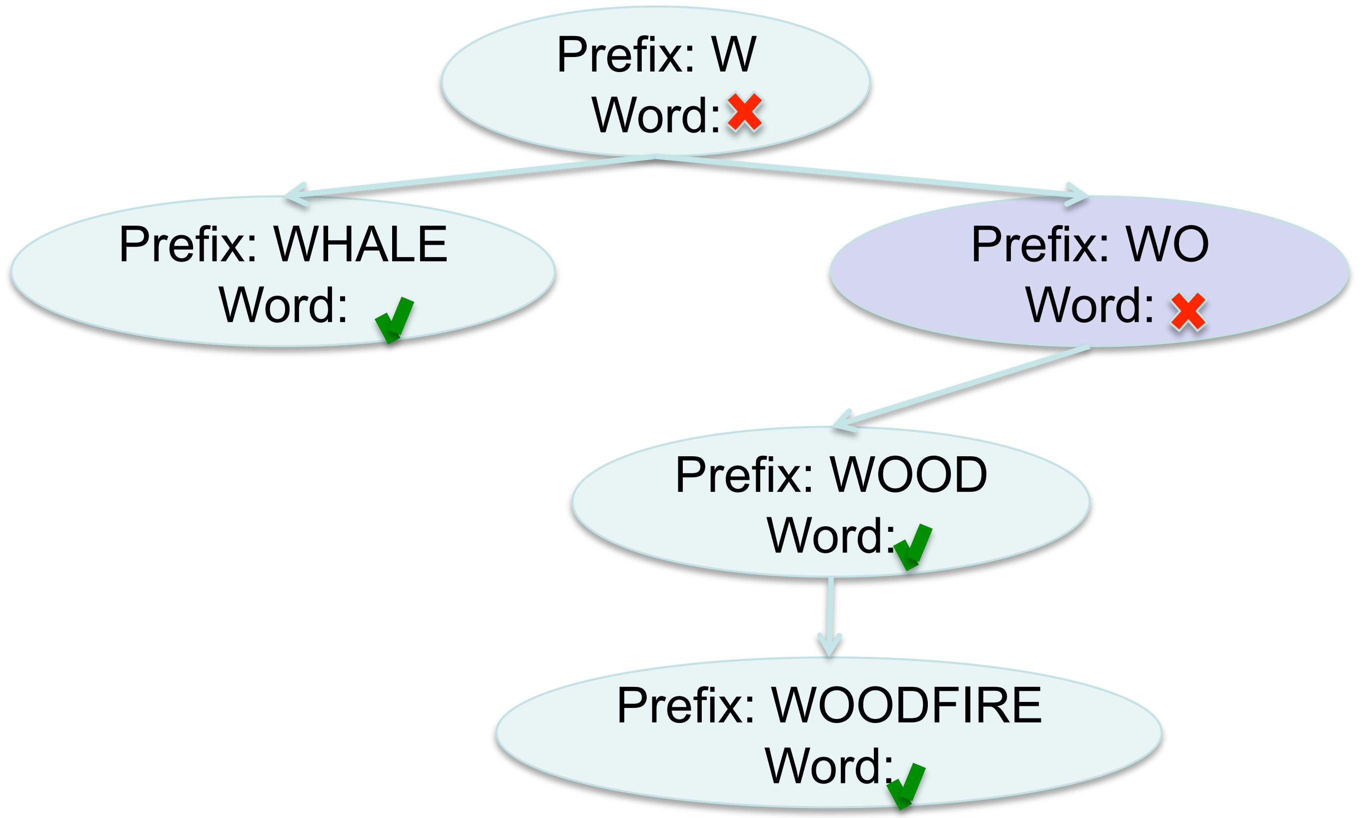
Insertion of WOLF (1)



Find longest Prefix with an existing String: WO

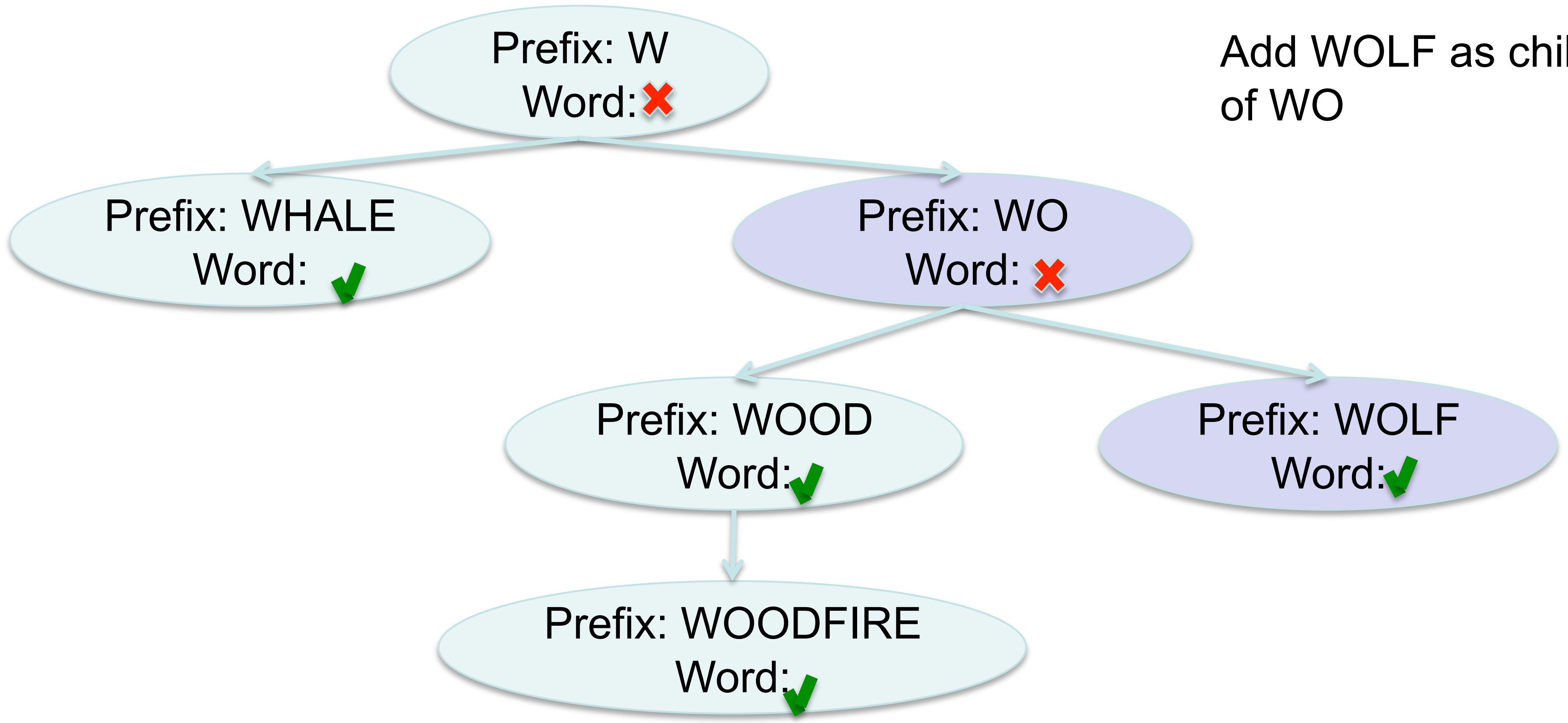
It is a part of the node WOOD

Insertion of WOLF (2)



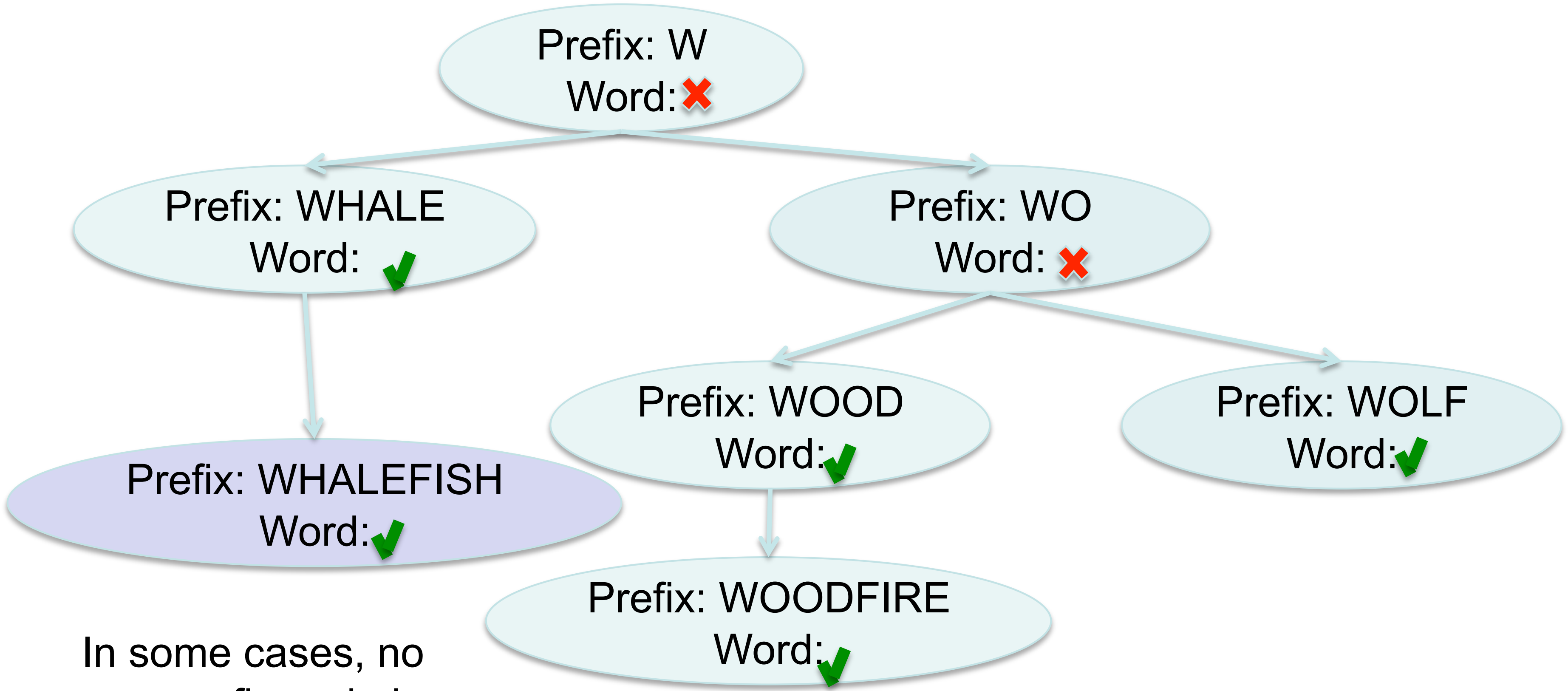
Insert a new node with the new common prefix

Insertion of WOLF (3)



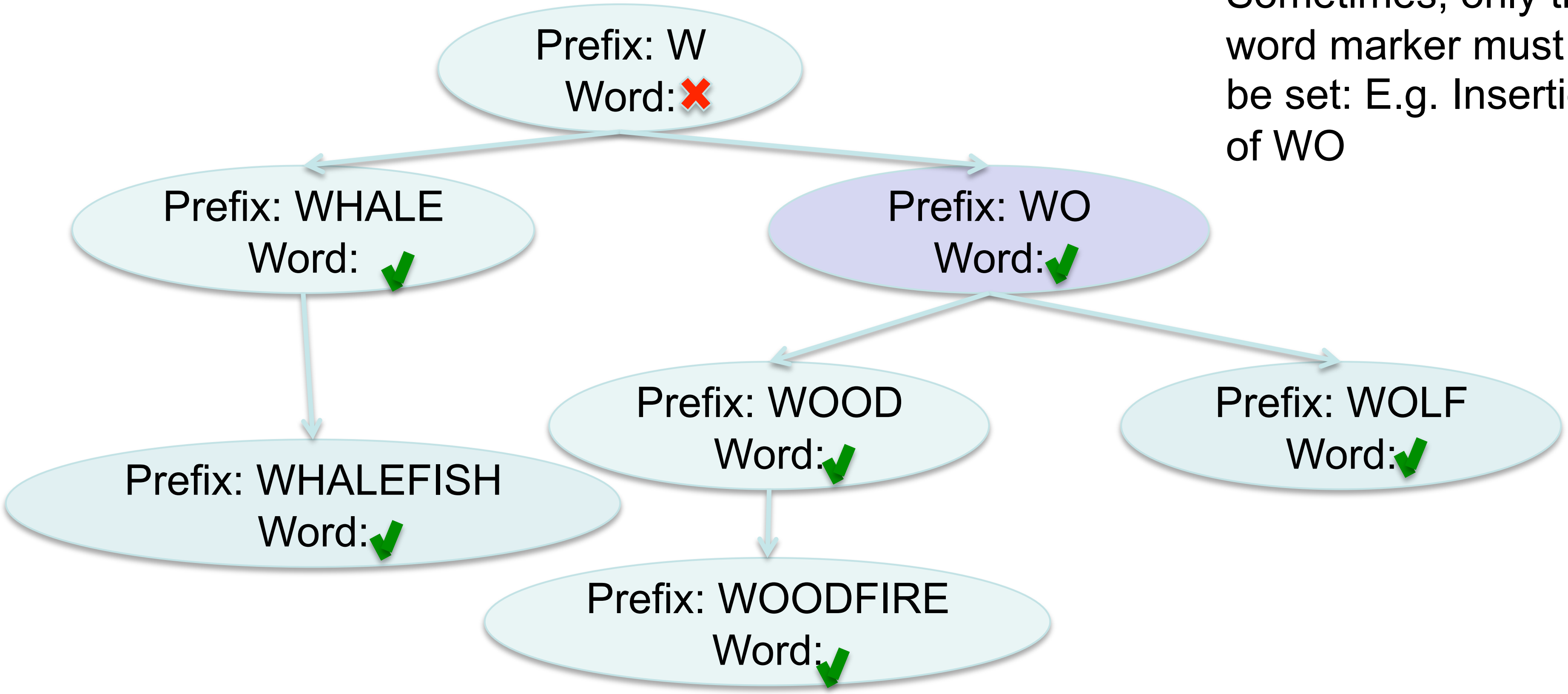
Add WOLF as child of WO

Insertion remarks (1)



In some cases, no new prefix node is needed

Insertion remarks (2)



Sometimes, only the word marker must be set: E.g. Insertion of WO