Technical Writing and Speaking in English TDTP 2: Ambiguity and Balance precision and clarity.

Anderson Andrei Da Silva and Quentin Guilloteau

Université Grenoble Alpes, Grenoble INP, Inria, LIG, France
email: danilo.carastan-dos-santos@inria.fr

August 2022

Exercise 1: For each sentence, write the possible meanings, and how to make the sentence non-ambiguous.

1. There is a man on a hill, and I am watching him with my telescope.
2. Look at the dog with one eye.
3. The duck is ready to eat.
4. Are you free tomorrow?
5. Visiting friends can be annoying.
6. Flying planes can be dangerous.
7. You should bring wine or beer and dessert.
8. She saw a bat.
9. He carries the light box.

Exercise 2: Based on the class, do you consider that the following excerpts are well written? If not, propose a modification.

1. Henry and Mark walked up to the door. He opened it.

2. Congrats to the best students, Alice and Bob.

3. Congrats to the best students, Alice, and Bob.

4. It is challenging to understand fog since it can get very complicated.

5. In terms of security, fog is very exposed since hackers can easily use fake, leading to the leak of important information.

6. For instance, if an end device generates data that must be handled but the end device's capacity makes processing the data impossible or impractical, the end device connects to a neighboring fog node, send the information to the fog node and offload the data processing to the fog node.

7. The low latency in-network transmission between the endpoint and the fog node, combined with the fog node's relatively high processing capacity, speeds up the entire process, allowing for real-time interactions.

8. The mathematical implementation of the node/link exceeding model is to put the values to the Fog function and equalize the nodes per link by setting the values to the process to reduce the number to 1, the base node.

9. Similarly, this same pattern is made for the Cloud server by putting the values for Cloud by keeping equivalent to nodes per exceeding links and gives the function y.

10. A simple implementation for a parallel merge sort is to simply create 2 tasks which execute a sequential merge sort on the two halves of the input array in parallel, and then merge the results of these two tasks.

11. This of course isn't feasible for big problem sizes since we would end up creating a lot more tasks than the cores/threads available to execute them.

12. The parallel bubble sort algorithm brings some overheads not only because of the thread creation and management done by OMP, but also because of the extra chunk border processing required.

13. On the basic parallelization, we create one task on every recursive call. To improve this, we decided not to add a task if the size is too small

14. For this algorithm we decided that the best option would be a recursive method. In order to achieve this, we added a conditional clause checking for the array size.

15. Bubble sort is one of the simplest sorting algorithm and with this simplicity comes difficulties to parallelize it.

Exercise 3: Write a small text (a few lines) about a topic, to be evaluated by a pair, in terms of clarity, ambiguity of the other texts:
Write your text here: `https://tinyurl.com/TWE15Sept`.
Evaluate the others here: `https://tinyurl.com/TWE15SeptPair`.
Submitt your correction here: `https://tinyurl.com/TWEWriteReviews`.
Check the review of your answer here: `https://tinyurl.com/TWESeeReviews`.