UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

Department of Computer Science
Faculty of Engineering, Built Environment & IT
University of Pretoria

COS226 - Concurrent systems

Practical 4 Specifications - Execution Tree Labelling

Release date: 26-08-2024 at 06:00
Due date: 30-08-2024 at 23:59
(FF will remain open until 01-09-2024 23:59 for technical difficulties)
Total marks: 62

Contents

1 General Instructions

- ***Read the entire assignment thoroughly before you start coding.***
- This assignment should be completed individually; no group effort is allowed.
- **To prevent plagiarism, every submission will be inspected with the help of dedicated software.**
- Be ready to upload your assignment well before the deadline, as no extension will be granted.
- If your code does not compile, you will be awarded a mark of zero. Only the output of your program will be considered for marks, but your code may be inspected for the presence or absence of certain prescribed features.
- If your code experiences a runtime error, you will be awarded a zero mark. Runtime errors are considered unsafe programming.
- Read the entire specification before you start coding.
- **Ensure your code compiles with Java 8**
- The usage of ChatGPT and other AI-Related software is strictly forbidden and will be considered as plagiarism.

2 Plagiarism

The Department of Computer Science considers plagiarism a serious offence. Disciplinary action will be taken against students who commit plagiarism. Plagiarism includes copying someone else's work without consent, copying a friend's work (even with consent), and copying material (such as text or program code) from the Internet. Copying will not be tolerated in this course. For a formal definition of plagiarism, the student is referred to http://www.library.up.ac.za/plagiarism/index.htm (from the main page of the University of Pretoria site, follow the Library quick link, and then choose the Plagiarism option under the Services menu). **If you have any form of question regarding this, please ask one of the lecturers to avoid any misunderstanding.** Also note that the OOP principle of code reuse does not mean that you should copy and adapt code to suit your solution.

3 Outcomes

On completion of this practical, you will have gained experience with the following:

- Execution Trees

4 Introduction

4.1 Execution Order labelling

In the context of binary consensus protocols, execution tree labelling refers to the classification of protocol states based on the potential outcomes they represent. A bivalent state is one where the decision value is not yet fixed; from this state, different execution paths can lead to different decision values (0 or 1). In contrast, a univalent state is one where the decision value is fixed, meaning all possible execution paths from this state will lead to the same decision value, either 0-valent or 1-valent. A critical state is a special type of bivalent state where any move by any thread results in a transition to a univalent state, thereby locking the decision value for the remainder of the protocol. Understanding these labels helps in analyzing the decision-making process within the protocol, especially in proving properties like wait-freedom and consensus.

5 Tasks

You are tasked with implementing a method to label all the nodes in an execution tree with the appropriate labels. Possible labels are `INITIAL, FINAL, UNIVALENT, BIVALENT, CRITICAL` these have been given in the labels.java file. The execution tree has been built for you, please do not change the code which builds the tree as FitchFork compares to this tree. You only have to label the nodes in the tree for this practical.

5.1 Task 1 - assignLabels

Implement the `assignLabels` method in the `ExecutionTree` class which assigns the correct labels to each node in the execution tree. Each node has a List of type Label, you must add to this list to assign a label to a node. The order of labels in this list is not important.
You are allowed to add methods and variables to the Node and ExecutionTree classes.

6 Upload checklist

The following files should be in the root of your archive

- `ExecutionTree.java`
- `Main.java` will be overridden
- `Label.java` will be overridden

7 Allowed libraries

These libraries will be allowed by the automaker but you must still do the given tasks with your own code (i.e you may not use the libraries to complete the task for you)

- `import java.util.*`

8 Submission

You need to submit your source files on the FitchFork website (https://ff.cs.up.ac.za/). All methods need to be implemented (or at least stubbed) before submission. Place the above-mentioned files in a zip named uXXXXXXXX.zip where XXXXXXXX is your student number. Your code must be able to be compiled with the Java 8 standard.

For this practical, you will have 20 upload opportunities and your best mark will be your final mark. Upload your archive to the appropriate slot on the FitchFork website well before the deadline. **No late submissions will be accepted!**