Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Parallel Programming
Assignment 12: Consensus
Spring Semester 2019

Assigned on: **20.05.2019** Due by: **27.05.2019**

Overview

Distributed consensus is a fundamental problem in computer science. It is defined as follows: There are N agents, each agent provides an input, i.e., an integer number. At the end of the consensus protocol, all agents need to agree on a single value out of the inputs provided by any agent. Some of the agents may fail or progress arbitrarily slow, yet we still require the protocol to terminate in a finite number of steps (wait-free consensus).

Exercise 1 – Wait-free implies lock free

Explain why a valid wait-free consensus protocol cannot use locks.

Exercise 2 – Valence states

Assume $N = 2$ and inputs are either 0 or 1 for each agent. Thus, in the initial state of any consensus, we are in a bivalent state (bivalent = the output can be 0 or 1). However, at termination, all agents have agreed on a single value, thus we are in a univalent state. Explain why there is a finite number of bivalent states in any wait-free consensus protocol.

Exercise 3 – Consensus among prisoners

Imagine there are 100 people in a prison. Each day the warden picks a prisoner (each prisoner with the probability 1/100). The prisoner is led to a room with a light that he can turn on or off. Initially the light is turned off. After the prisoner was in the room he can state ”by now every prisoner was in the room at least once”. If this statement is made and it is true, all prisoners are released. If the statement is made and it is false, all prisoners are shot. Devise a strategy that the prisoners can follow to make sure they get released some day in the future with absolute certainty (no other communication is allowed).

Exercise 4 – Implementing two thread consensus

Assume you have a machine with atomic registers and an atomic test-and-set operation with the following semantics (X is initialized to 1):

```
int TAS() {
  res = X;
  if (res = 1) {
    X = 0
  }
  return(res)
}
```

Implement a two-process consensus protocol using TAS() and atomic registers.

Exercise 5 – Linearizability

Which of the following scenarios are linearizable, assuming `s` is a stack? Either mark the point of linearization, or explain why the history is not linearizable.

a)
```
A : s.push(1)
B : s.push(2)
A : s.pop()
B : s.pop()
A : void
B : void
A : 2
B : 1
```

b)
```
A : s.push(1)
B : s.push(2)
A : void
B : void
A : s.pop()
A : 2
B : s.pop()
B : 1
```

c)
```
A : s.push(1)
B : s.push(2)
A : void
B : void
B : s.pop()
B : 1
A : s.pop()
A : 2
```

Submission

In order to receive feedback for your exercises, you need to submit your code to the Git repository. You will find detailed instructions on how to install and set-up Eclipse for use with Git in Exercise 1.

Once you have completed the skeleton, commit it to Git by following the steps described below. For the questions that require written answers, please write them on paper and bring them to the next exercise session where the solutions will be discussed.

- **Check-in your project for the first time**
  - Right click your created project called **assignment12**.
  - In the menu go to **Team**, then click **Share Project**.
  - You should see a dialog ”Configure Git Repository”. Here, next to the Repository input field click on **Create...**
  - Select a root git directory or your projects that you have created in Execise 1. Note for all your assignments you should use the same directory.
  - click **Finish**.
- **Commit changes in your project**
  - Now that your project is connected to your git repository, you need to make sure that every time you change your code or your report, at the end you commit your changes and send (push) them to the git server.
  - Right click your project called **assignment12**.
  - In the menu go to **Team**, then click **Commit...**.
  - In the Comment field, enter a comment that summarizes your changes.
  - In the Files list, select all the files that you changed and want them to be committed. This typically includes all the Java files but not necessarily all the files (e.g., you dont have to commit setting files of our eclipse installation).
  - Then, click on **Commit** to store the changes locally or **Commit and Push** to also upload them to the server. Note that in order to submit your solution you need to **both** commit and push your changes to the server.
- **Push changes to the git server**
  - Right click your project called **assignment12**.
  - In the menu go to **Team**, then click **Push Branch ’master’**. Note if this is not your fist push you can also use **Push to Upstream** to speed up the process.
  - A new dialog appears, now fill in for the URL field:
    `https://gitlab.inf.ethz.ch/COURSE-PPROG19/<nethz-username>.git`
  - Click **Next**
  - Keep the default values and click **Next**
  - An authentication dialog should appear. Fill in your nethz username and password and click **OK**.
  - Click **Finish** to confirm your changes. Note that eclipse might ask for authentication again.
- **Browse your repository online**
  - you can access and browse the files in your repository online on GitLab at:
    `https://gitlab.inf.ethz.ch/COURSE-PPROG19/<nethz-username>`