UiO : Department of Informatics
University of Oslo

IN5170 Models of Concurrency

Exam 2023

Author:
Kjetil K. Indrehus

November 22, 2024

Contents

1 General Questions

1.1 Problem 1: Interference and AMO

Consider the following program:

```
co
    <x:= x + y> // P1
||
    <y:= y + x> // P2
oc
```

Is the program Interference free?

To check it we can write the v and w variables:

$$v_{P1} = \{x, y\}, w_{P1} = \{x\}$$

$$v_{P2} = \{x, y\}, w_{P2} = \{y\}$$

Checking if the read and write interfere:

$$v_{P1} \wedge w_{P2} = \{y\} \neq \emptyset$$

$$v_{P2} \wedge w_{P1} = \{x\} \neq \emptyset$$

We see there are interference in the program, since neither union of read and write variables are empty set.

Does the two assignment satisfy AMO-property?

Since two assignments does not satisfy the AMO property since. We can verify this by using the AMO rule for assignments. It assigns a critical reference and it is also referenced in the other process. This means that the order of operations would lead to different results. It does not fulfills the AMO property.

1.2 Problem 2: fairness

Explain the difference between weak and strong fairness.

Solution

Fairness ensures that enabled statements should not be systematically be neglected by the scheduling strategy. We use conditions that are enabled or disabled. Both has to be *unconditional fair*. A scheduling strategy is unconditional fair if each enabled unconditional atomic action will eventually be chosen.
Weak fairness is when a condition gets enabled and stays enabled, then it will execute. Under *strong fairness* if a condition is infinitely enabled, then it will execute.

1.3 Problem 3: promise and future

Explain the difference between promises and futures.

Solution

Futures can be be though of as a mailbox that transmits return values. We need a way to identify the callback messages, and also a way to wait for the result. It is a handle for the caller of a process. It will contain the result value once computed. A future can be read multiple times, and can be used to synchronize with the calle.

Promise is a future that has is not clear who is computing it. A promise is only completed once.

1.4 Problem 4: future and linear channels

1.5 Problem 5: channels