Assignment 1: Introduction and challenges of concurrency

ETH Zurich

1 Amdahl's Law

1.1 Background

Consider a program where multiple threads operate on a buffer. Some threads only read from the buffer and other threads only write to the buffer. Any number of readers can simultaneously operate on the buffer. While a writer is operating on the buffer, no other writer or reader can be active on the buffer.

Assume a pool of N threads where each reader and writer is a thread. Hereby, 90 % of the threads are readers and 10 % of the threads are writers. Each reader thread takes 2 seconds to execute and each writer thread takes 3 seconds to execute. The program terminates when all threads in the thread pool terminated.

1.2 Task

According to Amdahl's Law, what is an upper bound for the speedup of the above implementation on a 4-core processor?

1.3 Solution

$$S = \frac{1}{1 - p + \frac{p}{n}}$$

$$p = \frac{0.9 * 2}{0.9 * 2 + 0.1 * 3}$$

With $n = 4$ this results in:

$$S = \frac{14}{5} = 2.8$$

2 Interleavings

2.1 Background

This exercise is taken from the book *Principles of Concurrent and Distributed Programming* [2]. Imagine two threads P and Q that share the variables K and n.

$n := 0$			
P		Q	
1	**do** K **times**	1	**do** K **times**
2	$temp := n$	2	$temp := n$
3	$n := temp + 1$	3	$n := temp - 1$

2.2 Task

What are the possible final values of n for a given positive value of K?

2.3 Solution

The final value of n can be any value between $-K$ and K. There are two main cases. Either P executes one loop iteration between the moment Q reads n in line 2 and the moment Q is about to write to n in line 3. In this case Q will overwrite the effect of P's iteration. In the other case the roles of P and Q are swapped. If the first case occurs over and over again then we end up with $n = -K$. In the second case the execution results in $n = K$. The other possible combinations of the two cases result in the values between $-K$ and K.

3 LTL Models

3.1 Background

The transition system $\mathcal{M}$ in Figure 3.1 models a barbershop with one barber, two customers, and one chair for waiting. Each customer i can be in four different states: $entering_i$ upon entering the shop, $waiting_i$ when waiting for the barber, $haircut_i$ when getting a haircut, and $gone_i$ upon leaving the shop. A waiting customer can leave the shop when the barber is busy, and a customer on the barber chair can switch with a waiting customer.
The following notation is used in the formulas:

G	Globally	F	Future
X	Next	U	Until
$\rightarrow$	Implication	$\wedge$	Conjunction
$\vee$	Disjunction	$\neg$	Negation

3.2 Task

Answer the following questions about the formulas given in linear-time temporal logic (LTL). Justify your answers in each case.

1. Let ϕ be $(F\ gone_1) \wedge (F\ gone_2)$.
   (a) Is ϕ satisfied at state q_8, i.e., $\mathcal{M}, q_8 \models \phi$?
   (b) Is ϕ satisfied globally, i.e., $\mathcal{M}, q \models \phi$ for all states q?
2. Let ϕ be $waiting_1\ U\ haircut_1$.
   (a) Is ϕ satisfied at state q_6, i.e., $\mathcal{M}, q_6 \models \phi$?
3. Let ϕ be $(entering_1 \wedge XX\ haircut_1) \rightarrow X\ haircut_1$.
   (a) Is ϕ satisfied at state q_1, i.e., $\mathcal{M}, q_1 \models \phi$?
   (b) Is ϕ satisfied globally, i.e., $\mathcal{M}, q \models \phi$ for all states q?

Translate each of the following properties from natural language into LTL.

4. No two customers get a haircut at the same time.
5. If customer 1 enters the barber shop, (s)he will leave the shop eventually.
6. Before leaving the barber shop, customer 2 is always either waiting or getting a haircut.
7. Once customer 1's haircut is done, (s)he leaves immediately.

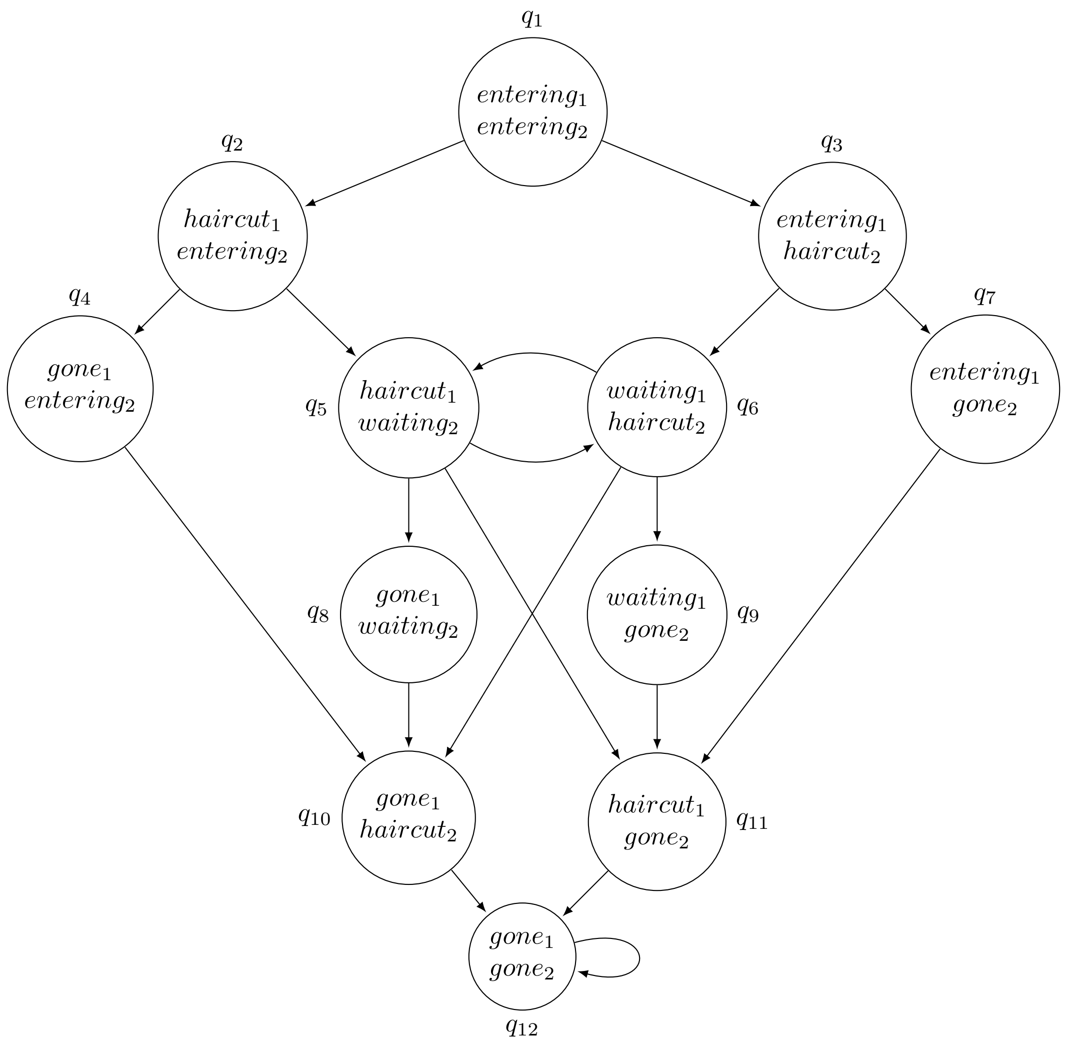
q_1 $entering_1$ $entering_2$
q_2 $haircut_1$ $entering_2$
q_3 $entering_1$ $haircut_2$
q_4 $gone_1$ $entering_2$
q_5 $haircut_1$ $waiting_2$
q_6 $waiting_1$ $haircut_2$
q_7 $entering_1$ $gone_2$
q_8 $gone_1$ $waiting_2$
q_9 $waiting_1$ $gone_2$
q_{10} $gone_1$ $haircut_2$
q_{11} $haircut_1$ $gone_2$
q_{12} $gone_1$ $gone_2$

Figure 1: Barbershop model

3.3 Solution

1. (a) Yes, all paths go through state q_{12} which satisfies $gone_1$ and $gone_2$.
   (b) No, some paths never satisfy $gone_1$ or $gone_2$, e.g., $(q_1, q_2, q_5, q_6, q_5, \dots)$.
2. (a) No, e.g., $(q_6, q_{10}, q_{12}, q_{12}, \dots)$
3. (a) Yes, all the paths from q_1 satisfy the formula.
   (b) No, e.g., $(q_3, q_7, q_{11}, q_{12}, q_{12}, \dots)$.
4. $G(\neg(haircut_1 \wedge haircut_2))$
5. $G(entering_1 \rightarrow F\ gone_1)$
6. $G((waiting_2 \vee haircut_2)\ U\ gone_2)$
7. $G(haircut_1 \rightarrow X\ (haircut_1 \vee gone_1))$

4 Safety vs. liveness

4.1 Task

Consider the following properties.

1. What goes up must come down.
2. If two or more processes are waiting to enter their critical sections, at least one succeeds.
3. If an interrupt occurs, then a message is printed.
4. The cost of living never decreases.
5. Two things are certain: death and taxes.
6. You can always tell a Harvard man.

For each of the above properties, state whether it is a safety or liveness property. Identify the bad or good thing of interest.

4.2 Solution

With respect to the order provided in the question, the answers are as follows.

1. This is a liveness property. The good thing to happen is that the thing having gone up is coming down.
2. This is a liveness property. The good thing to happen is that at least one processor succeeds.
3. This is a liveness property. The good thing to happen is the message being printed.
4. This is a safety property. The "bad" thing that will never happen is that the cost of living decreases.
5. This is a liveness property. The "good" thing to happen is death and taxes.
6. This is a safety property. The bad thing that will never happen is that you cannot tell a Harvard man.

5 Interleavings in practice

5.1 Background

We know that the interleavings in a concurrent program may give rise to different behavior. This exercise is designed to give a way to see how unpredictable these effects may be.

5.2 Task

Your task is to design a Haiku composer. A Haiku is a Japanese form of poetry with 17 syllables in three lines, where the first line must contain 5 syllables, the second must contain 7, and the third line must contain 5 (this is the traditional layout). The lines may contain any number of words, as long as the syllable restrictions are followed. The Haiku composer will have a small (20-30 should be enough) list of words, and will spawn 3 threads to compose a Haiku poem. Each thread is responsible for a single line of the Haiku.

For this task, you must use a single shared store of words. Once a thread has used a word, it must be removed from the store. You may find the usage of the **java.util.concurrent** package helpful here. The store should have a reasonable number of 1-3 syllable words. It is also perfectly OK to keep removing words until you find the one that "fits" your syllable requirement. You may wish to define a **Word** class which can model a word, including syllable count.

This should be done without using concurrency operations such as **synchronized** and the **wait/notify** capabilities of objects.

To spawn threads and the basics of java concurrency, you may refer to the chapter on Java concurrency in the course book available at
http://se.inf.ethz.ch/courses/2014a_spring/ccc/reading_materials/book/.

5.3 Solution

The solution is available in the source code that comes with this solution.

References

[1] Andrei Voronkov. Script to Logic in Computer Science. 2009.

[2] Mordechai Ben-Ari. Principles of Concurrent and Distributed Programming (2nd Edition). Addison-Wesley, 2006.