Concurrency:
State Models & Design Patterns

Q&A Session

Week 13
Exercises 12

Discussion
Exercise 12 - Task 1

Answer the following questions:

a) What is a Software Architecture? What is its benefit?

A Software Architecture defines a system in terms of computational components and interactions amongst those components. It breaks the overall complexity down to several smaller blocks that are easier to handle. This improves the maintainability and ultimately the modularity.

b) What are the potential disadvantages when using layered architectures?

It increases the development overhead in small projects and it restricts the freedom of choice in terms of coding styles.
Exercise 12 - Task 1

Answer the following questions:

c) Provide an example in which the pattern Specialist Parallelism could be a legitimate architectural choice. Justify your answer!

Specialist Parallelism makes very much sense for different workloads that require different computation strategies, as initiated by web-servers (networking, data analysis and management, storage, ...).

d) The concepts Result Parallelism, Specialist Parallelism and Agenda Parallelism represent three ways of thinking about the problem. Can you tell on what they focus? Provide one sentence for each one of them.

Result Parallelism focuses on the shape of the finished product, Specialist P. on the makeup of the work crew and Agenda P. on the list of tasks to be performed.
Exercise 12 - Task 1

**Answer the following questions:**

**e)** What is a Flow Architecture? What are Blackboard Architectures?

In Flow Architecture architectural patterns (e.g. Unix pipes) synchronization is ensured by the linear processing in which information only flows in one direction from sources over filters to sinks. Blackboard Architecture architectural patterns (e.g. Producer/Consumer) perform all synchronization in a “coordination medium” where agents can exchange messages.

**f)** Which blackboard style should be preferred when we have multiple processors? Why?

The Agenda Parallelism should be selected, as it allows you to instantiate as many threads as CPU cores exist. The tasks thereof are typically independent and the workers identical.
Exercise 12 - Task 1

Answer the following questions:

g) What are Unix pipes and how do you use them?

Unix pipes (established through the “|” symbol) are bounded buffers that connect producer and consumer processes. They can be used in shell commands (e.g. `ls -f | wc -l`) to “connect” different input / output streams together to easily perform complex data manipulations.
Exercise 12 - Task 2

Now we change roles: It is your turn to send me questions of topics you are still not familiar with for the next practical session. The best questions will be presented in front of you. I will try my best to answer all the questions you submit. For each question you ask you will retrieve a point (maximum of 3 points, no bonus this time :).
Exercise 12 - Task 2

7 questions  Safety / Liveness / Fairness
8 questions  FSP
6 questions  Synchronization Techniques
5 questions  Petri Nets
2 questions  Conceptual
6 questions  Language Specifics
3 questions  Concurrency Architectures
1 question  Organizational Affairs

Totally 38 Questions in 8 Categories
Exercise 12 - Task 2

Safety / Liveness / Fairness
Q:
What is the difference between safeness and liveness?

A:
Short: Safeness is related to corruption (logical error states / crashes). Liveness is related to progress and deadlocks.
Long: Both can relate to each other as deadlocks could be a result of corruptions.
Safety can be verified with traversal graphs. Liveness cannot be verified easily. (Halting problem)
Exercise 12 - Task 2 - Q002

Q:
How would you manually check a safety property?

A:
- Is an error state reachable?
  (- Is the error state logically correct?)

A = (a -> x -> ERROR).
Q: What is the difference between tracking state and using state-tracking variables?

A: It's just about activeness/passiveness.

Tracking state (passive approach):
We get notified when external state changes (somewhere).

Tracking state variables (active approach):
We observe a (local) state variable; if it changes we notify appropriate components ourselves.
How would you manually detect a waits-for cycle and manually check a progress property?

**Waits-for cycle:**
Graph construction, apply fancy algos on graph

**Progress-property:**
Ping all processes and use response timeouts to determine if progress happened within a time slot
Exercise 12 - Task 2 - Q005

Q:
What are the dangers in letting the (Java) scheduler choose which writer may enter a critical section?

A:
It is biased by the randomness of the system pseudo-random-number-generator (PRNG). However, if the implementation is sound no issues (deadlocks) must occur, but starvation could still happen.
Exercise 12 - Task 2 - Q006

Q:
What assumptions do nested monitors violate?

A:
They especially could violate Liveness assumptions. (deadlock, nested monitor lockout)
Q:
Liveness and Guarded Methods, slides 4 and 5: What is the difference between liveness and progress property? Are starvation, dormancy, premature termination and deadlock all livelocks?

A:
Progress has stronger constraints than liveness. You can model most liveness properties as progress properties.

"Most" means not all: [enter] -> [random] -> [exit] is not possible in progress properties, as you have to define all processes by name.

Exercise 12 - Task 2 - Q007 #2

Q:
Liveness and Guarded Methods, slides 4 and 5: What is the difference between liveness and progress property? Are starvation, dormancy, premature termination and deadlock all livelocks?

A:
No, livelock is a special case of resource starvation. All involved processes **constantly change states but achieve no progress**. For example, two processes constantly try to let the other run to achieve some progress. Both would end up to only change the active process, but do no progress at all.
Exercise 12 - Task 2
Q:
I’m still a little bit confused about the parallel composition operator “||”. How does it work exactly?

A:
Step 1: Search common actions of involved processes
Step 2: Start building graph with common action(s)
Step 3: Rebuild before / after action(s) of both processes
Step 4: Repeat step 3 until all potential actions are included

see next slide ...
Exercise 12 - Task 2 - Q008
Exercise 12 - Task 2 - Q009

Q:
The example below is from the LTSA documentation. [PIC1.png] What I don’t understand are the two unlabeled transitions from 0 to 2 and from 2 to 3. Otherwise the example would perfectly make sense for me (since we always need an “a” and a “b” action before we can execute the “x” action).

A:

see next slide ...
Exercise 12 - Task 2 - Q009

Parallel Composition "||"

\{ P || Q \} expresses the parallel composition of the processes \( P \) and \( Q \). It constructs an LTS which allows all the possible interleavings of the actions of the two processes. Actions which occur in the alphabets of both \( P \) and \( Q \) constrain the interleaving since these actions must be carried out by both of the processes at the same time. These shared actions synchronise the execution of the two processes. If the processes contain no shared actions then the composite state machine will describe all interleavings.

In the following example, \( x \) is an action shared by the processes \( A \) and \( B \).

\[
A = \{ a \rightarrow x \rightarrow a \}, \quad B = \{ b \rightarrow x \rightarrow b \},
\]

\[
||SYS = (A || B).
\]

A:

Flawed diagram. Cannot be reproduced.
Q:
To what extend are we expected to write down FSPs in the exam? Can you provide some more sample questions to practice on? Does the syntax have to be 100% correct in the exam?

A:
You should be able to understand and adapt the FSPs seen in the exercises, as some FSPs can be easily adapted to other environments/parameters. You should have a look at such details. Syntax has to be 100% correct for all the points (but I will perhaps introduce an "all-inclusive" deduction for whatever syntactical errors exist).
Q:
Can you explain the calculation of traces in FSP again?

A:
That highly depends on the FSP. You should try to map it onto a different structure (tree, graph, grid, ...) and calculate the possibilities based on reliable algorithms.

The example based on Race5K can be easily mapped to a grid. X-axis represents tortoise turns, y-axis hare turns. Then all paths from bottom left to top right have to be found.
Exercise 12 - Task 2 - Q012

Q: How can we model general Blackboard Architecture case using FSP?

A: This is just a high-level conceptual implementation, with no concurrency support!

\[ \text{WORKER}_A = (\text{getA} \rightarrow \text{thinkA} \rightarrow \text{putA} \rightarrow \text{WORKER}_A). \]
\[ \text{WORKER}_B = (\text{getB} \rightarrow \text{thinkB} \rightarrow \text{putB} \rightarrow \text{WORKER}_B). \]

\[ \text{BLACKBOARD} = (\text{getA} \rightarrow \text{putA} \rightarrow \text{BLACKBOARD} | \text{getB} \rightarrow \text{putB} \rightarrow \text{BLACKBOARD}). \]
Exercise 12 - Task 2 - Q013

Q:
Consider the following FSP, how do you guarantee fairness?
COIN = (toss -> heads -> COIN | toss -> tails -> COIN ).

A:
It is a high-level concept that has to be implemented yourself in FSP.
FAIRNESS = (tossH -> heads -> tossT -> tails -> FAIRNESS).

You can verify it with these statements:
progress HEADS = {heads}
progress TAILS = {tails}
Q:
Series 04, Ex 3: I do not see, why value 8 (9?) for i is not allowed? Because LIFT[8] will lead to LIFT[9] and this is an action within property, so it should be ok... of course I have checked the LTSA and I have seen that it is not the case... but I am just confused why not...

A:
You specify the bounds of i with LIFT[i:0..8]. So i cannot take the value 9.
Q: Series 07 (05?), Ex 4: What does the 7 in the last line (GET OUT = MAZE(7)) mean? I have changed the number, but the LTSA stays the same...

A: When you compose the MAZE "7" is the start state for the composition.
Exercise 12 - Task 2

Synchronization Techniques
Exercise 12 - Task 2 - Q016

Q:
How would we implement fair semaphores? With a FIFO queue into which threads are put according to their order of arrival and taken out the same way?

A:
Exactly! That's one valid solution.
Q:
Besides of its disadvantages (it causes starvation) is there any advantage of busy-wait (except it is easy to implement)?

A:
No. However, in some programming languages you may have no other options.
Exercise 12 - Task 2 - Q018

Q:
Is there any case in which using notifyAll() should be avoided?

A:
Not in general. Speed improvements could be a reason for avoiding notifyAll(), but that's only important in REALLY large-scale systems.
Exercise 12 - Task 2 - Q019

Q:
Could you again explain "implement the monitor using semaphore" and vice-versa example? I think a lot of students (me included) had some difficulties when we first discussed the solution and I'm still not 100% sure if I'd do it correctly. I'd be nice to look back at it now that we have some more skills :)

A:
see next slide
public class Semaphore {
    private int s = 1;
    // P(s), delay until s > 0
    public synchronized void enter() {
        while (s <= 0) {
            try {
                wait();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        s = s - 1;
    }
    // V(s), executes s := s + 1
    public synchronized void leave() {
        s = s + 1;
        notifyAll();
    }
    ...
}
Monitor with Semaphores

public class Monitor {
    Lock lock = new BinarySemaphore(0); // can also be ordinary semaphore
    Lock lock2 = new BinarySemaphore(1); // can also be ordinary semaphore

    public void wait() {
        lock.acquire(1);
    }

    public void notify() {
        lock2.acquire(1); // MUTEX:
        if (lock.hasQueuedThreads()) {
            lock.release(1); // only one lock has to be released per condition
        }
        lock2.release(1); // evaluation
    }
}
Exercise 12 - Task 2 - Q020

Q:
Are binary semaphores as good as counting semaphores?

A:
Binary semaphores are less flexible than counting semaphores. Binary semaphores are like on/off switches whereas counting semaphores can maintain various states (e.g. serve up to 10 clients, ...).
Q:
Is it better when message passing is synchronous or asynchronous? (I think it depends on the circumstance where we use message passing, since both synchronous and asynchronous have their individual pros and cons, but not sure with my answer).

A:
It depends on the use case, but since it often includes another transfer medium (network, ...) it is in many cases mandatory to be asynchronous.
Exercise 12 - Task 2
Q:
Does mean “all transitions live” that all transitions are live at once or each of the transitions can be live at one time?

A:

**Definition of Liveness:**
A transition is live if it can never deadlock.

So transitions are either live or not. It is *not* dependent on some firing events. It is given by the Petri and its initial marking.
Q:
Should we able to write the regular expression of the defined petri net? Is this important? Or is it enough for us to know how to define a petri net?

A:
No, the ability to derive regular expressions from a Petri net is not necessary.
It is important to know how to define a Petri net, though.
Exercise 12 - Task 2 - Q024

Q: Petri Nets, slide 5: For me the marking is rather a set than a function. Why is it defined as a function from places to Nat?

A:  
1) It is not a set because it can contain the same item multiple times.  
2) It should be a function, but relay on another input value:
   
   \[ m: C \rightarrow \text{List of places} \]
Q: How can a (bad) implementation of a Petri net deadlock even though there are enabled transitions?

A: Definition deadlock:
A transition is deadlocked if it can never fire.
So a Petri net cannot deadlock as long as there are any enabled transitions, but some individual transitions possibly can.
Exercise 12 - Task 2 - Q026

Q:
What constraints could you put on a Petri net to make it fair?

A:
There are no easy constraints. You need to model this high-level concept on your own. Similar to the FSP example.
Exercise 12 - Task 2
Exercise 12 - Task 2 - Q027

Q:
With the buzz of asynchronous programming, is it still a good idea to assign one thread per client (for example in the context of an HTTP server)? Why?

A:
It makes sense for smaller workloads. If the workloads increase, it may make sense to split the work caused by each client to multiple threads to handle the work most efficiently.
There exist production-grade web servers using either one or the other approach.
Q: In what tasks using multithreading is justified? What number of threads should we choose for the particular task?

A: It is justified for all tasks for which the expected speed gain is worth it. The number of threads really correlates with the tasks and the system. Rule of thumb: To achieve maximum efficiency establish a 1:1 relationship between threads and physical CPU cores.
Exercise 12 - Task 2

Language Specifics
Q:
In Java, are there primitive data types that are inherently thread-safe, for example the int type? If yes, why are there concurrent counterparts like the AtomicInteger data type designed explicitly for use in concurrent applications?

A:
This originates from the CPU registers, because old CPUs could only write one 32 bit value at a time. Consequently all <= 32 bit primitive values are thread-safe (int, byte, boolean, ...), the larger ones > 32 bit are not (long, double).
Out of convenience for inexperienced developers?
Exercise 12 - Task 2 - Q030

Q:
In the exercises we were expected to answer some Java specific questions. I had to google most of the stuff since I don't have much experience in concurrent Java programming. To what extend should we prepare Java concurrency nitty-gritties for the exam?

A:
We don't ask any Java nitty-gritties in the exam, but of course, you should know the concepts and their application, benefits, ....
Exercise 12 - Task 2 - Q031

Q:
Why is the Java Slot class so much more complex than the FSP Slot specification?

A:
You just see the benefit of highly specialized languages. They can achieve very compact code, but this code is usually hard to interpret. Another important factor is the feature availability. LTS does not have a fully-fledged type (int, String, ...) or visibility (public, private, ...) system, but Java does.
Exercise 12 - Task 2 - Q032

Q:
What is the use of class invariants and how/why are they typically used in Java?

A:
They are used as reference holders for immutable object references or immutable values (configuration parameters).
Q: In the slides CP-03-Safety on page 10 (15?). What does adding VarAlpha to the Inc process actually do and especially how?

A: It's an alphabet extension. It is sometimes useful to extend the alphabet of a process with actions that it does not engage in and consequently actions that are not used in its definition. This may be done to prevent another process executing the action.
Exercise 12 - Task 2 - Q034

Q:
When should you use synchronize(this) rather than synchronize(someObject)?

A:
You should always use synchronize(someObject). When you use synchronized(this) other code could steal your monitor through accidental synchronized usages in the class.
Exercise 12 - Task 2

Concurrency Architectures
Exercise 12 - Task 2 - Q035

Q:
In which case we should prefer Flow Architecture over Blackboard Architecture and vice versa?

A:
Blackboard introduces more overhead, but is more flexible. In the end it's a tradeoff between modularity, scalability and speed.
I would prefer the flow architecture for more constrained local problems (Unix pipe like) and the blackboard architecture for big data preparation and processing.
Q: Can Futures be applied in Agenda or Specialist Parallelism to speed up?

A: Technically yes, but does that make any sense?
As long as the result is not yet calculated, the blackboard doesn't need to be aware of that, because the worker himself will publish the work as soon as it gets ready. Even worse, when the worker dies, the ticket would be lost as well, so that wouldn't improve anything, too. This would just add an additional layer of unnecessary complexity.
Exercise 12 - Task 2 - Q037

Q:
In which situations does Specialized Parallelism make sense?

A:
In all situations where we can achieve a high benefit from highly specialized workers. Imagine hardware accelerated GPUs that support only matrix calculations. Then a specific type of workers would only be responsible for these calculations. All the other tasks would be processed by other workers that do not rely on GPUs.
Exercise 12 - Task 2

Organizational Affairs
Exercise 12 - Task 2 - Q038

Q:
Which materials are important for the exam?

A:
Lecture slides, exercises, Git code samples, Petri net examples, this slides.
Exercises about slides of chapter 13 (actors) or the lab won't appear in the exam.
... still more questions?
Next Time: Exam
60 minutes / CLOSED BOOK

20-Dec-2017, 10:15am until 11:15am

1. Arrive on time

2. Don't forget your student ID

3. Don't forget a blue or black ball pen

4. Not allowed: pencils, internet, notebook, books, any printouts, additional blank pages, pocket calculators, mobiles, smart anything, ...
Good Luck!