7 Formal Specifications - exam questions

You should know the answers to these questions

Why is an UML class diagram a semi-formal specification?

A UML class diagram is a semi-formal specification because it uses a precise syntax for structure and relationships but lacks strict formal semantics, leaving room for interpretation.

What is an automated theorem prover?

An automated theorem prover is a tool that uses formal logic to verify that specific properties or post-conditions are satisfied given the preconditions and code structure.

What is the distinction between "partially correct" and "totally correct"?

- **Partially Correct**: If the precondition holds and the function terminates, the postcondition is true.
- **Totally Correct**: The function is guaranteed to terminate, and when it does, the postcondition is true, assuming the precondition holds.

Give the mathematical definition for the weakest precondition of Hoare triple {P} S {Q}.

The weakest precondition $wp(S, Q)$ for a statement S and postcondition Q is defined as the least restrictive condition P such that, if P holds before S, then Q will hold after S.

Why is it necessary to complement sequence diagrams with statecharts?

Sequence diagrams depict specific scenarios or message exchanges, while statecharts describe all possible states and transitions, ensuring completeness and capturing system dynamics comprehensively.

What is the notation for the start and termination state on a state-chart? What is the notation for a guard expression on an event?

- **Start State**: Represented by a filled circle.
- **Termination State**: Represented by a filled circle with an enclosing circle.
- **Guard Expression**: Written in square brackets `[]` adjacent to the transition.

What does it mean for a statechart to be (a) consistent, (b) complete, and (c) unambiguous?

- **Consistent**: Every state is reachable from the initial state, and all final states are reachable from every other state.
- **Complete**: Every possible event-state pair has a defined transition.
- **Unambiguous**: No event or guard expression appears more than once leaving the same state.

How does a formal specification contribute to the correctness of a given system?

Formal specifications allow mathematical proof of system properties, ensuring correctness and aiding in traceability from requirements to implementation. They also facilitate comprehensive test case generation.

You should be able to complete the following tasks

Use a theorem prover (Daphny) to prove that a given piece of code is correct.

Ensure preconditions, loop invariants, and postconditions are specified for the code. Use Daphny's logic verification tools to prove correctness.

Create a statechart specification for a given problem.

Define states, transitions, events, and guard conditions. Use statechart conventions (start/termination states, nested states, etc.) to represent the system's behavior.

Given a statechart specification, derive a test model using path testing.

Construct test cases to cover every state transition, validating behavior against predicates and edge cases.

Can you answer the following questions?

(Based on the article "A Formal Approach to Constructing Secure Air Vehicle Software".) What is according to you the most effective means to achieve "provably secure against cyberattacks"?

The most effective approach combines formal verification of the architecture and component properties with synthesis of software using secure programming languages. Using formally verified micro-kernels also ensures robust separation and communication constraints.

Why is it likely that you will encounter formal specifications?

Formal specifications are crucial for high-risk systems, embedded systems, and standardized protocols due to their reliability, traceability, and capability to prevent catastrophic failures.

Explain why we need both the loop variant and the loop invariant for proving total correctness of a loop?

- **Loop Invariant** ensures correctness by holding true at initialization and after every iteration.
- **Loop Variant** ensures termination by strictly decreasing with every iteration and reaching a boundary condition.

What do you think happened with the bug report on the broken Java.utils.Collection.sort()? Why do you think this happened?

The bug was rooted in the failure to maintain the algorithm's invariant. Though fixed in many projects, OpenJDK opted for a suboptimal fix (increasing array lengths), lacking machine-verified proof, possibly due to resource constraints or risk assessment.

Explain the relationship between "Design By Contract" on the one hand and "State-based specifications" on the other hand.

- **Design By Contract**: Defines preconditions, postconditions, and invariants for operations.
- **State-based Specifications**: Define acceptable states and transitions, inherently specifying preconditions and postconditions for state changes.

Explain the relationship between "Testing" on the one hand and "State-based specifications" on the other hand.

State-based specifications guide testing by identifying all possible state transitions and guard conditions, enabling comprehensive test coverage for system behaviors.

You are part of a team building a fleet management system for drones transporting medical goods between hospitals. You must secure the system against cyber-attacks. Your boss asks you to look into formal specs; which ones would you advise and why?

I would recommend:

- **Input/Output Specifications**: For specifying security-critical preconditions and postconditions.
- **State-Based Specifications**: To model drone states, transitions, and error handling.
- **Formal Verification**: To mathematically prove system security properties, ensuring robustness against attacks.