CS132: Software Engineering

Midterm Exam

8:15-10:00, May 11th, 2023

There are 7 problem sets and the total points are 20 points. Each problem set includes a few questions. For each question, the maximum possible points are stated.

Please write your answers **legibly on the answer booklet** so that we can read and understand your answers. If a problem seems ambiguous, please feel free to state your assumption explicitly and solve the problem. Obviously, your assumption should be reasonable and should not trivialize the problem.

Pledge. Copy the following pledge and sign your name in your answer booklet:

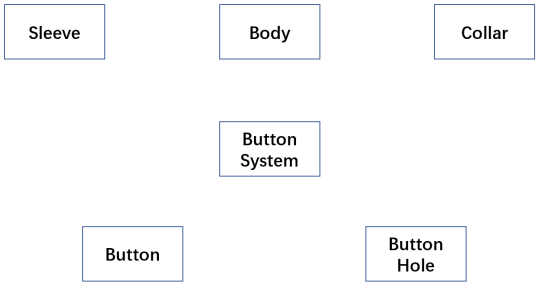
I neither cheated myself nor helped anyone cheat on this exam.

Problem 1. (4 points)

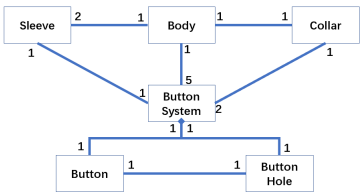
In this question, you are asked to draw a class diagram for a shirt. A shirt has one body, two sleeves and one collar. There are multiple button systems on the shirt, each consists of a button and a button hole. Each sleeve has one button system. Each body has 5 button systems. Each collar has 2 button systems.

(2 points) Please draw the association relationships of the components on the figure below.

(2 points) Please indicate the multiplicity on **each** association relationship (i.e. how many Component A corresponds to how many Component B).

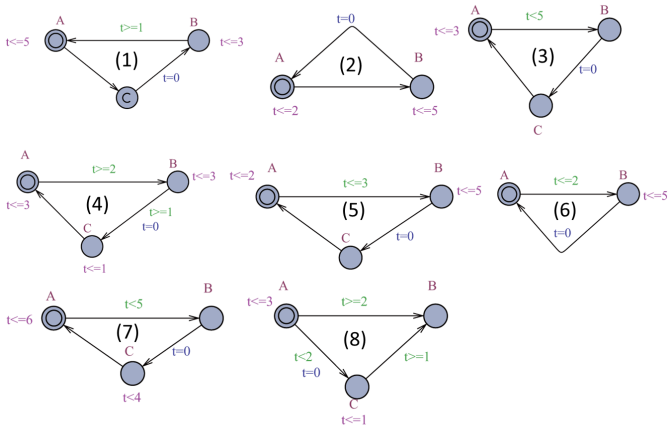


Answer:



Problem 2. (2 points)

Assume t is a defined clock, there are 8 different UPPAAL models shown in the figure below. Please answer the following questions.



(2 points) For each model, write down whether the model has deadlock. For the model(s) you think have deadlock, please explain the reason for deadlock.
Hint: A model has deadlock when there is no enabled transitions at certain state during execution.

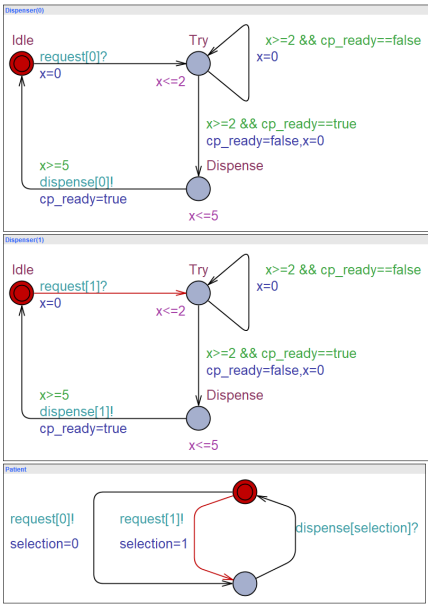
0.5 for each yes/no answer. -0.2 if correctly answered Deadlock but wrong explanation.

1. **Answer:** No deadlock.
2. **Answer:** No deadlock.
3. **Answer:** Deadlock. C to A may violate invariant of A

4. **Answer:** No Deadlock.
5. **Answer:** Deadlock. C to A may violate invariant of A
6. **Answer:** Deadlock. May stay in A over 2 time steps
7. **Answer:** Deadlock. No enabled transition when t==6
8. **Answer:** Deadlock. No enabled transition in B

Problem 3. (3 points)

The following UPPAAL model models an automatic mutiple-type drug dispenser.



The Dispenser template models the behavior of the dispense logic for one type of drugs. All dispensers uses a common compartment to contain and deliver the drug to the patient. Whether the compartment is ready for use is modeled by the variable *cp_ready* (true: ready for use; false: being used by some dispenser). So we have to make sure that at any time, there is only one dispenser using the compartment.

The *intended* behavior for Dispenser is that, upon getting a patient request with drug type (0 or 1), the corresponding dispenser process tries to wait for 2 time units and check if *cp_ready*. If yes,

it gains access to the compartment for exactly 5 time units to dispense the drug (a physical process not modeled), notifies the patient, releases the compartment access, and waits for the next request; otherwise, it waits another 2 time units and repeat the check. The Patient template models the behavior of a patient requesting drugs from the Dispenser template and waiting for the type of drug to be dispensed.

(a) **(2 points)** Suppose you want to check that the dispenser algorithm is correct by checking the following properties using UPPAAL. State each of the following properties as a UPPAAL query.

  (a) The system does not have deadlock
  Answer: A[] not deadlock. True.

  (b) For Dispenser with drug type 0 (referred to as Dispenser 0 in the sequel), when it tries to obtain access to the compartment, it will eventually obtain the access. [Hint: To express that Dispenser 0 is in the location IDLE, use the syntax "Dispenser(0).IDLE".]
  Answer: Dispenser(0).Try → Dispenser(0).Dispense, or A◇ Dispenser(0).Try imply Dispenser(0).Dispense, True

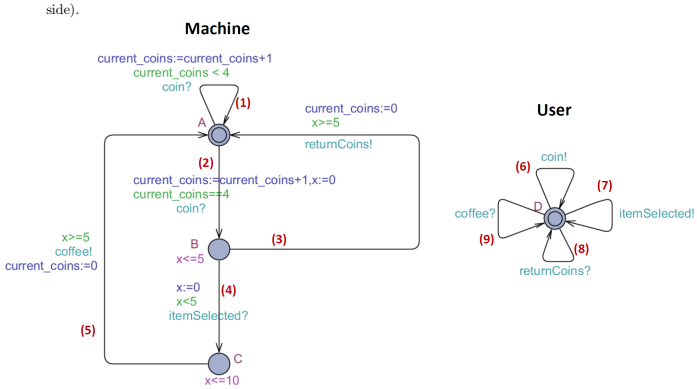
  (c) Dispenser 1 and Dispenser 2 can never be using the compartment at the same time.
  Answer: A[] not(Dispenser(0).Dispense && Dispenser(1).Dispense), True

(b) **(1 points)** For each property in Part (a), state whether it is true or false with regards to the given model and the textual description (a supposedly correct UPPAAL query, not necessarily the query you wrote). And provide the explanations.

Problem 4. (3 points)

Coffee machine is a system that interacts with users in order to serve various types of coffee. Users insert coins and select preferred items, then the coffee machine serves the selected coffee to the user.

The following UPPAAL model abstracts the behavior of (coffee) machine (left-side) and user (right-side).



The machine (left-side) has two variables:

- current_coins: an integer variable to keep track of the number of coins user inserted; the initial value is 0.
- x: clock variable.

The machine has 3 locations:

- In location A, the machine waits until the user provides sufficient coins through the channel **coin**.
- In location B, the machine waits until the user selects preferred coffee through channel **itemSelect**; if the user selects it within 5 time units, the machine takes a transition to location C; otherwise, the machine takes a transition to location A by returning the inserted coins, which is implied through the channel **returnCoins**.

- In location C, the machine prepares coffee, and serves it to the user, which is implied through the channel **coffee**.

Answer the following questions.

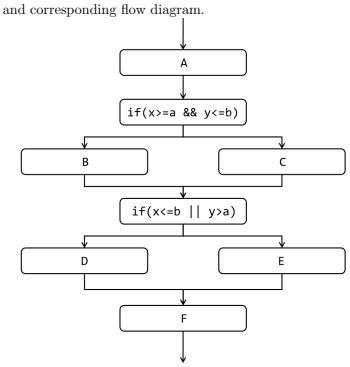
(a) **(1 points)** Does the following property hold? If not, explain with a counter example. If yes, justify.
A<> Machine.C
Not satisfied. The system may stay in location A forever.

(b) **(2 points)** Does the following property hold? If not, explain with a counter example. If yes, justify.
A[] (Machine.A imply (Machine.current_coins≤100))
Satisfied. The variable current_coins will be less than 5 in Machine.A

Problem 5. (3 points)

Consider the following C program snippet and corresponding flow diagram.

```
int main(){
    int x, y, a, b;
    ... // Input x & y
    a = 4;
    b = 10;                                   A
    if(x >= a && y <= b){
        x = x + y + a;
        y = y - a + b;                        B
    }
    else{
        x = x / a + b;
        y = y * b;                            C
    }
    if(x <= b || y > a){
        ...                                   D
    }
    else{
        ...                                   E
    }
    ...
    return 0;                                 F
}
```



(a) **(1.5 points)** Please specify a test suite that achieves branch coverage with the minimum number of test cases and write down the path for each test case.

Answer: Branch coverage:
$x = 10, y = -2, A \to B \to E \to F$
$x = 1, y = 10, A \to C \to D \to F$

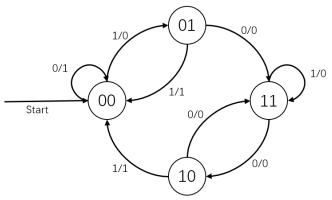
(b) **(1.5 points)** Please specify a test suite that achieves condition coverage with the minimum number of test cases and write down the path for each test case.

Answer: Condition coverage:
$x = 10, y = -2, A \to B \to E \to F$

Example: $x = 1, y = 10, A \to C \to D \to F$

Problem 6. (3 points)

A state machine is shown below. 00, 01 are the name of the state, and input/output are marked on the transitions.



Answer the following questions.

(a) **(1 points)** Complete the State Table for the state machine.

State/Input		

(b) **(2 points)** Identify 0-switch test cases for the state machine and calculate the coverage of your test suite.

$$\text{Coverage} = (\frac{N}{T} \times 100\%)$$

In which N is the number of test coverage items covered by test cases, and T is the number of identified test coverage items.

Test case								
Start state								
Input								
Final state								
Expected output								
Test coverage item								

Problem 7. (2 points)

Consider the following MATLAB application adopted from the starter template for the projects.

This application, named as "counter", maintains a count as its state. When the "+" button is clicked, the count will increase by one. And when the "-" button is clicked, the count will decrease by one. The label on the UI will be updated accordingly.
Please fill in the empty lines below to make the counter functioning as described.
Hint: **1**. The *controller* property in the app is **private**, which cannot be accessed directly. Read the code carefully for solutions. **2**. You may use *num2str* to convert *int32* to *str*.

main.m:

```
% Entrypoint for counter.

% Initialize the app and the controller.
app = view();
_____________________________________
_____________________________________
```

answer:

```
% Entrypoint for counter.

% Initialize the app and the controller.
app = view();
controller = counter(app);_____________
app.constructor(controller);___________
```


counter.m:

```
% Create a counter object
% @property {int} count - The current count
classdef counter < handle

    properties (SetAccess = private)
        count int32
        app  view
    end

    methods

        % @constructor
        % @param {view} app - A reference to the app
        % @return {Self} self
        function self = counter(app)
            self.app = app;
            self.count = 0;
        end

        % Increment the counter
        % @param {Self} self
        % @return {int} count - The current count
        function count = increment(self)
            _____________________________________
            _____________________________________
            count = self.count;
            return;
        end

        % Decrement the counter
        % @param {Self} self
        % @return {int} count - The current count
        function count = decrement(self)
            _____________________________________
            _____________________________________
            count = self.count;
            return;
        end

    end

end
```

answer:

```
% Create a counter object
% @property {int} count - The current count
classdef counter < handle

    properties (SetAccess = private)
```

```
        count int32
        app   view
    end

    methods

        % @constructor
        % @param {view} app - A reference to the app
        % @return {Self} self
        function self = counter(app)
            self.app = app;
            self.count = 0;
        end

        % Increment the counter
        % @param {Self} self
        % @return {int} count - The current count
        function count = increment(self)
            self.count = self.count + 1;
            self.app.Label.Text = self.count + "";
            count = self.count;
            return;
        end

        % Decrement the counter
        % @param {Self} self
        % @return {int} count - The current count
        function count = decrement(self)
            self.count = self.count - 1;
            self.app.Label.Text = self.count + "";
            count = self.count;
            return;
        end

    end

end
```

view.mlapp:

```
classdef view < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        CounterUIFigure matlab.ui.Figure
        Image           matlab.ui.control.Image
        ButtonMinus     matlab.ui.control.Button
        ButtonPlus      matlab.ui.control.Button
        Label           matlab.ui.control.Label
    end

```



```


    properties (Access = private)
        % @property {counter} controller: Controller for the counter
        controller
    end

    methods (Access = public)

        % @contructor
        % @param {counter} controller - A reference to the controller
        function constructor(app, controller)
            app.controller = controller;
        end
    end


    % Callbacks that handle component events
    methods (Access = private)

        % Button pushed function: ButtonPlus
        function ButtonPlusPushed(app, event)
            app.controller.increment();
        end

        % Button pushed function: ButtonMinus
        function ButtonMinusPushed(app, event)
            app.controller.decrement();
        end
    end

    ...

end
```