Embedded Systems

Duration : 2h **All documents allowed**

The three parts are independent. Please answer on 3 separate sheets.
Informal explanations in plain english will be appreciated a lot, and it is compulsory to justify all answers.
The number of points associated with each question is only an indication and might be changed slightly.

Part I - Modeling Time and Concurrency (5 points)

We consider programs that make use of a "counter" semaphore initialized with 3.

The atomic operations on such a semaphore S are the following:
– P(S) decrements the counter
– Z(S) tests whether the counter is 0,

There are 3 processes P_1, P_2, P_3. The process P_i executes the sequence of code: `Before`$_i$ `; P(S) ; while not Z(S) do nothing ; After`$_i$.

`Before`$_i$ and `After`$_i$ are considered to be atomic. The processes are executed on a monoprocessor machine, with a preemptive scheduler.

▷ **Question 1 (4 points) :**
— Describe each of the processes by an automaton, taking into account the definition of atomicity.
— Explain how you build their asynchronous product to represent the set of all possible behaviors of the system.

▷ **Question 2 (1 points) :**
— What global property is ensured by the use of the counter semaphore?
— Is it a safety, or a liveness property?

Part II - Abstract Interpretation (5 points)

Let us consider programs manipulating integer variables in the domain $[0, +\infty[$. We would like to perform an analysis of these programs with intervals, but not *all* intervals. We consider only the intervals with bounds in $\{0, 2, 10, +\infty\}$. Let us denote by $\mathcal{A}$ the set of these intervals.

▷ **Question 3 (2 points) :**
Give all the elements of $\mathcal{A}$, and organize them into a lattice (an interval I is "less" than another interval J in this lattice, if and only if $I \subseteq J$).

We consider two program points `A, B`, and transitions between them (in the control graph, or interpreted automaton).

▷ **Question 4 (3 points) :**
— If there is a transition labeled by `x := x*2` from `A` to `B`, and we know that the possible values of `x` at point `A` are abstracted by the interval $I \in \mathcal{A}$, by which interval in $\mathcal{A}$ can you abstract the values of `x` at point `B`?
— If there is a transition labeled by `x > 2*y` from `A` to `B`, and we know that the possible values of `x, y` at point `A` are abstracted respectively by the intervals $I_x, I_y \in \mathcal{A}$, by which intervals in $\mathcal{A}$ can you abstract the values of `x, y` at point `B`? Examine only 4 cases, and explain why they are significant.

Part III - Model-Checking (10 points)

1 Explicit versus Symbolic Representation

Consider the Verilog module shown in Figure 1.

▷ **Question 5 (5 points) :**

1. Draw the Kripke structure defined by this module. Which states satisfy the formula EX(`vld=1` ∧ `data_out=2'b00`)? (Recall that `2'b00` just means that two bits are `0`. So, `data_out=2'b00` says that both bits of the register `data_out` have to be `0`).
2. Write an initial and a transition predicate for the module. Give a predicate over the state variables that characterizes all states satisfying `data_out=2'b00` ∧ EX(`data_out=2'b01`).

```
module bridge(clk, en, data_in, vld, data_out);
   input clk;
   input en;
   input  [1:0] data_in;
   output vld;
   output [1:0] data_out;

   reg  vld;
   reg [1:0]  data_out;

   initial begin
      vld <= 0;
      data_out <= 2'b00;
   end

     always @(posedge clk) begin
        if (en && (data_out <= data_in)) begin
           vld <= 1;
           data_out <= data_in;
        end else begin
           vld <= 0;
        end
        if (data_out == 2'b11) begin
           data_out <= 2'b00;
        en
     end
   endmodule // bridge
```

Figure 1: An implementation of a simple bridge.

2 Specifying using Logic

Assume two atomic propositions a and b. Write logical formulas describing the following properties over these two propositions.

▷ **Question 6 (2 points) :**

1. Write a CTL formula stating that from every state there is a path to a state in which a is satisfied.
2. Write an LTL formula stating that a and b are never true at the same time.
3. Give a CTL formula stating that from the initial state, we can eventually reach a state satisfying a.
4. Write an LTL formula describing all the traces on which a is true infinitely often or b is false infinitely often.

3 Binary Decision Diagrams

Note that, as usual, we use the shortcut BDD to refer to a Reduced-Ordered Binary Decision Diagram.

▷ **Question 7 (3 points) :**

1. Draw the BDD representing the Boolean expression $(u \wedge \neg v) \wedge ((u \wedge x) \vee \neg w \vee \neg y)$ under the variable ordering $u \prec v \prec w \prec x \prec y$ (i.e., u is the top-level variable) and give the BDD node table. Recall a node table consists of a column for the node name, one for the variable name, one for the 0-successor node, and one for 1-successor node.