Instructions to Candidates

• Answer ALL questions.

• Answers must be written in the space provided in this booklet; otherwise they will not be graded.

• All answers MUST come with the correct explanations. There is no credit for guessing. A correct answer without the correct explanation will receive no marks.

• This is an OPEN BOOK examination. You are allowed to bring in any books/lecture notes etc.

• You can ask for extra sheets for rough work.

• PLEASE WRITE YOUR MATRICULATION NUMBER BELOW.

MATRICULATION NO.: 

(This portion is reserved for the examiner's use only)

<table>
<thead>
<tr>
<th>Question</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question A</td>
<td>3</td>
</tr>
<tr>
<td>Question B</td>
<td>4</td>
</tr>
<tr>
<td>Question C</td>
<td>3</td>
</tr>
<tr>
<td>Question D</td>
<td>3</td>
</tr>
<tr>
<td>Question E</td>
<td>3</td>
</tr>
<tr>
<td>Question F</td>
<td>2</td>
</tr>
<tr>
<td>Question G</td>
<td>2</td>
</tr>
<tr>
<td>TOTAL</td>
<td>20</td>
</tr>
</tbody>
</table>
A. 3 marks

Are the following Linear Time Temporal Logic (LTL) formulae equivalent? If yes, give a proof. If not, construct examples to show that they are not equivalent.

\[ \neg FGp \text{ and } G(\neg p \lor XF\neg p) \]

You can assume that \( p \) is an atomic proposition.

**Answer:** The two formulae are equivalent. Since \( Fp \) is satisfied by a trace \( \pi \) iff (a) either the first state of \( \pi \) satisfies \( p \), or (b) one of the second or later states of \( \pi \) satisfies \( p \) this results in the following equivalence

\[ Fp = p \lor XFp \]

Now, \( \neg FGp = GF\neg p \). Using the above equivalence of \( Fp \) we get

\[ \neg FGp = GF\neg p = G(\neg p \lor XF\neg p). \]
B. 4 marks Consider the following program with two processes, which are composed asynchronously. Assume that initially $x == y == 0$.

\[
x = 1 \quad a = x \\
y = 1 \quad b = y
\]

What are the possible values of $a$ and $b$ when the program terminates? For each of these possible values draw a trace that will generate it.

Answer:

1. $a == b == 0$
   
   \[
   a = x \\
b = y \\
x = 1 \\
y = 1
   \]

2. $a == b == 1$
   
   \[
   x = 1 \\
y = 1 \\
a = x \\
b = y
   \]

3. $a == 0, b == 1$
   
   \[
   a = x \\
x = 1 \\
y = 1 \\
b = y
   \]

4. $a == 1, b == 0$
   
   \[
   x = 1 \\
a = x \\
b = y \\
y = 1
   \]
C. 3 marks

Are the two following Linear time Temporal Logic (LTL) formula equivalent? If yes, give a proof. If not, construct example traces to show that they are not equivalent. You can assume that \( \varphi \) is an arbitrary LTL formula.

\[ G(\varphi \Rightarrow X\varphi) \]

\[ G(\varphi \Rightarrow G\varphi) \]

**Answer:** Consider a path \( \pi \) satisfying \( G(\varphi \Rightarrow X\varphi) \). Let \( k \) be an index \( \geq 0 \) such that \( \pi^k \models \varphi \) where \( \pi^k \) denotes the suffix of \( \pi \) starting from position \( k \). Then, clearly \( \pi^{k+1} \models \varphi \). Again this means \( \pi^{k+2} \models \varphi \). A simple induction on \( i \) is able to establish that for any natural number \( i \) we must have \( \pi^{k+i} \models \varphi \). This means \( \pi \models G(\varphi \Rightarrow G\varphi) \).

The proof in the other direction is trivial and follows from the definition of \( G \) and \( X \) operators. If a path satisfies \( G(\varphi \Rightarrow G\varphi) \) — let \( k \) be an index \( \geq 0 \) such that \( \pi^k \models \varphi \) where \( \pi^k \) denotes the suffix of \( \pi \) starting from position \( k \). Then, clearly \( \pi^{k+1} \models \varphi \). Thus, \( \pi \models G(\varphi \Rightarrow X\varphi) \).
D. 3 marks

Consider a system consisting of temperature controller, a thermostat, an air-conditioning unit and a heater unit. When the controller receives temperature-high signal from the thermostat, it sends an on signal to the air-conditioning unit, and an off signal to the heater unit. When the controller receives temperature-low signal from the thermostat, it sends an on signal to the heater unit, and an off signal to the air-conditioning unit. If the controller receives a normal signal from the thermostat, it turns off both units.

Construct multiple Sequence Diagrams showing sample behaviors of the above-mentioned reactive system. You can only get full credit if your collection of Sequence Diagrams is detailed enough to cover as much of the above requirements as possible.

Answer:
E. 3 marks

Construct the overall behavior of the system in Question D as one single UML State Diagram. The thermostat can be treated as external environment. All other components are considered to be part of the “system”.

Answer:
F. 2 marks

Suppose we want to verify the LTL formula $G(p \Rightarrow Fq)$, for a concurrent system $Sys$ where $p,q$ are atomic propositions. As per the LTL model checking algorithm discussed in class, a property automata will be synchronously composed with the state machine of $Sys$. What is the property automata that will be synchronously composed with the state machine of $Sys$ in this case?

Answer:

The negation of the property is $\neg(G(p \Rightarrow Fq))$, that is, $\neg G(\neg p \vee Fq)$, that is, $F(\neg(\neg p \vee Fq))$, that is, $F(p \land \neg Fq)$, that is, $F(p \land G(\neg q))$. 

![Diagram](image-url)
G. 2 marks

```c
int x = 0;
while(1){
    x = x + 1;
}
```

Can you use model checking to prove the LTL property $F(x = 649)$. Why or why not?

**Answer:** The conventional answer is "no" - because model checking is restricted to finite state systems. Thus, if we try to construct the state space of the above program it will not terminate since the value of $x$ comes from an unbounded domain (integer) and also $x$ does assume unboundedly many values.

One can try to argue (with quite a stretch) that the answer is "yes". This is the case, if we do not try to construct the state space prior to traversal - but rather construct and traverse it on-the-fly. In this case, we can have a bounded representation of $x$ (say 32 bits) – and construct the state space during its traversal. In this case, we will encounter the value $x = 649$ as we are constructing/traversing the state space. However, we still need to reason explicitly that this program has only one execution trace and that is why when we encounter $x = 649$ in that execution trace, our proof is complete. This piece of reasoning is, strictly speaking, not being done inside the model checking search which is only trying to search for counter-examples rather than proofs.