uOttawa

L'Université canadienne
Canada's university

SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING

COURSE: SEG2106
SEMESTER: WINTER 2011

PROFESSOR: Gregor v. Bochmann
DATE: February 19, 2011
TIME: 9:00 to 10:30

MIDTERM EXAMINATION

Solutions

NAME and STUDENT NUMBER:______________________________/________________

Mid-Term Exam

There are three (3) types of questions in this examination.

Part 1	**Multiple choice and simple answers**	**18 marks**	
Part 2	**Short development questions**	**37 marks**	
Part 3	**Problem solving**	**10 marks**	
Total		**65 marks**	

The space allocated for each question is limited. In case of necessity you may use the other side of the pages to continue.

Note: There is limited time. Do not spend too much time on the questions that give only two or three points.

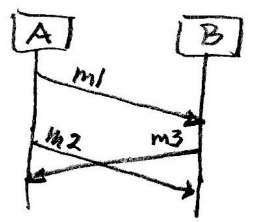
Annex A: Example of Lex input
Annex B: Example of LTSA model specification and tool output
Annex C: Algorithm 3.3 (Thompson)

Multiple choice and short-answer questions:

1. **[2 marks]** Sometimes one defines "verification" as answering the question "Do we build the system right ?", and "validation" as answering the question "Do we build the right system ?" . Explain in a few words what is meant by "verification".

   Verification means to check that the implementation, or the system design conforms to the requirements. It is assumed here that the requirements have been documented precisely. (It is not necessary to consult with the stakeholders for doing "verification")

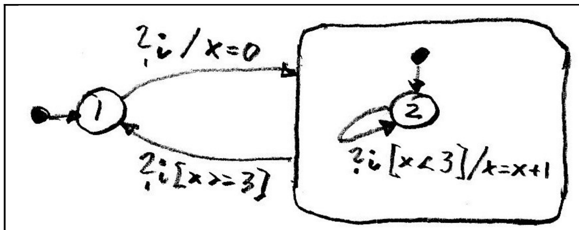
2. **[3 marks]** The sequence diagram below shows 6 events related to the exchange of three messages between the entities A and B. Please explain in a few words (using as example the sequence diagram below) why one associates often a partial order with the events in a sequence diagram.



Answer: The diagram defines a partial order between events because certain pairs of events are not ordered, such as the sending of m2 and the sending of m3. The diagram defines the order between all local events for each system component identified, and clearly for each message, the reception is after the sending.

3. **[3 marks]** The diagram below shows a UML State Diagram, where i is an input, x is an integer variable, and State 2 contains a entry action that generates an output o with an integer parameter equal to the current value of x.

   Write down the sequence of input and output interactions that would be generated if the state machines receives 8 consecutive i inputs.



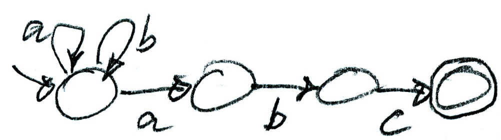
Answer:

input	Value of x	output
i	0	O(0)
i	1	O(1)
i	2	O(2)
i	3	O(3)
i	3	
i	0	O(0)
i	1	O(1)
i	2	O(2)

4. **[2 marks]** Do the following two regular expressions define the same language ?
   - (a+ b+)* c
   - (a b) (a* b*)* c

   Answer: (a) YES (b) **NO**

5. **[2 marks]** Does the following regular expression define the same language as the accepting automaton shown here ? - regular expression: (a | b) (a | b)* c



Answer: (a) YES (b) **NO**

6. **[3 marks]** Annex A contains the text of a file destined as input to the LEX (or Flex) tool. Explain in a few words, for each of the following lines of code, what the meaning of the code on that line is.
Line 23: STUDENT_ID has the following form: starts with u or s, then followed by 6 or 7 digits.
Line 27: When STUDENT_ID is encountered the variable students is incremented.

If you used JavaCC in the Lab (instead of LEX or Flex), you may instead answer the following question: How would you write in JavaCC the definition of a regular expression that defines the set of sequences of the following form: one or two characters “a” followed by a “c” or “b”, and then followed by zero or more occurrences of the sequences “abc” or “bba”.

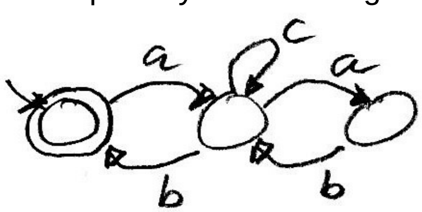
(a | aa) (c | b) (abc | bba)*

7. **[3 marks]** Annex B contains the specification of a labeled transition system (LTS) – a simple coffee machine - in the input format of the LTSA tool. It also contains the output produced by that tool when the command “CHECK” -- “progress” was executed. – Explain in a few words: (a) the meaning of the output produced by the tool, and (b) what is the nature of the problem contained in the given LTS specification.

(a) The tool output states that there is a cycle including the interaction tryToFix and that this cycle can be reached after a specific trace of interactions (coin.10, etc.). Such a loop can be called a livelock.
(b) If after the indicated trace the interaction tryToFix is executed, the system remains in a state where tryToFix is the only interaction that can be executed.

Short development questions

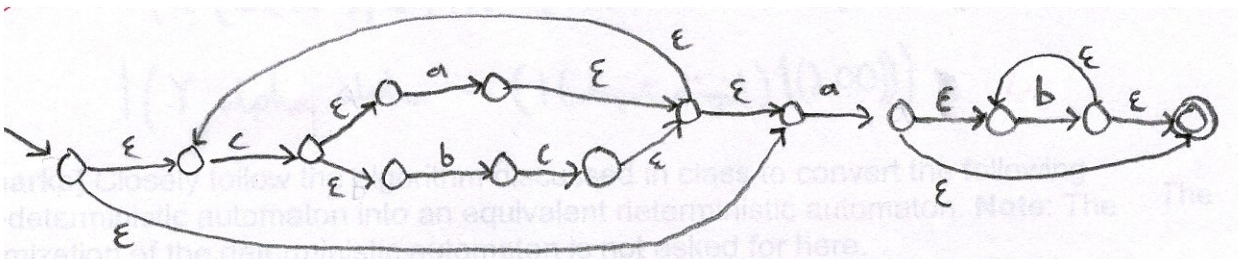
8. **[4 marks]** Write down a regular expression that defines the regular language accepted by the following automaton:



Answer: **(a (c | ab)* b)***

9. **[6 marks]** Closely follow the algorithm of Thomson (see Algorithm 3.3 in Appendix A if needed) to construct a non-deterministic automaton that accepts the language defined by the regular expression “ (c (a | b c)) * a b* “

Note: I asked for **closely** following Thomson’s algorithm.

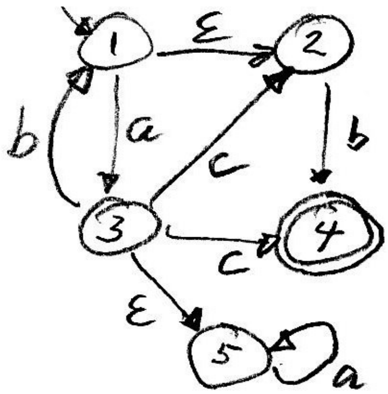


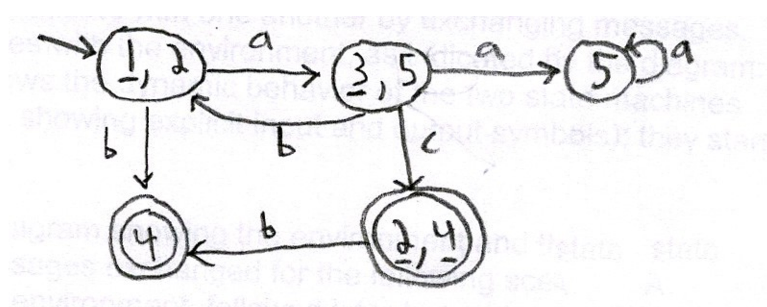
10. **[4 marks]** The account numbers of the customers of a bank have the following form: The number either starts with an "X" followed by a number of two digits between 08 and 33, followed by "-" and then by 2 to 3 decimal digits, or it starts with a "Y" and 2 alphabetic characters followed by "-" and decimal value between 100 and 200 (inclusively). – Write down a regular expression that defines the language of all valid account numbers of this bank. You may assume that the following definitions are given:

    digit = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
    alpha = A | B | C | … | Z

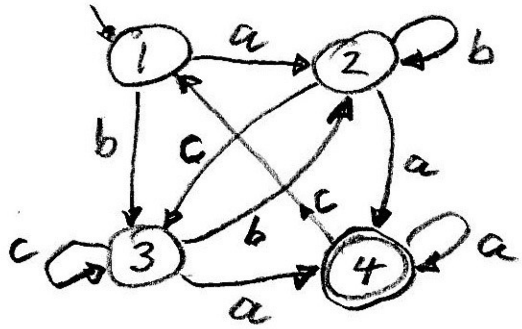
    Answer: X (0 (8|9) | (1|2) digit | 3 (0|1|2|3)) – digit digit digit?
    | Y alpha alpha – (1 digit digit | 200)

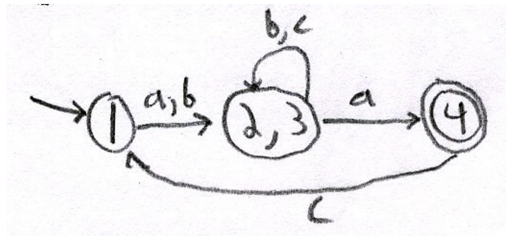
11. **[8 marks]** Closely follow the algorithm discussed in class to convert the following non-deterministic automaton into an equivalent deterministic automaton. **Note**: The minimization of the deterministic automaton is not asked for here.





12. **[3 marks]** Find the minimal automaton equivalent to the following automaton.

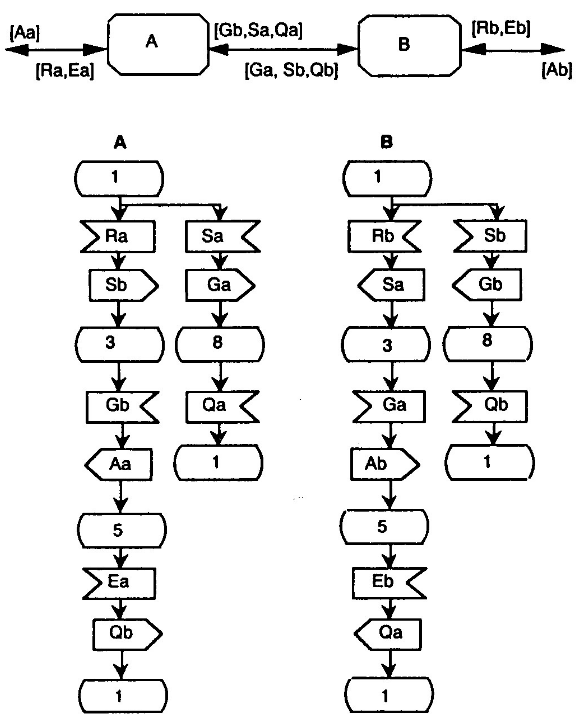


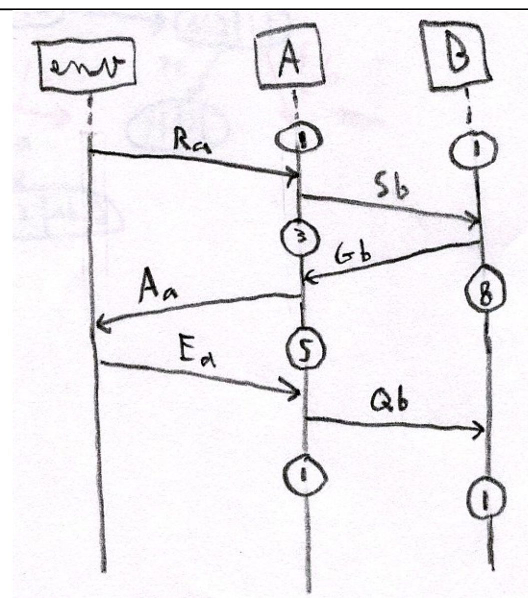


13. **[6 marks]** The diagram below shows an architecture diagram showing two state machines A and B that communicate with one another by exchanging messages. They also exchange messages with the environment, as indicated by the diagram. In addition, the figure below shows the dynamic behavior of the two state machines (using the UML-SDL notation showing explicit input and output symbols); they start out in their respective state 1.

(a) Write down a sequence diagram showing the environment and the two state machines, as well as the messages exchanged for the following scenario: A message Ra arrives from the environment, followed later by a message Ea (after a message Aa has been received by the environment).

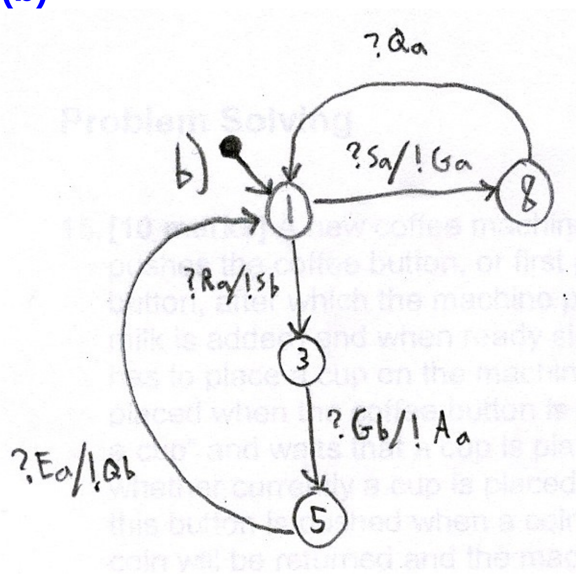
(b) Write down a UML State Diagram (using the notation like in the example of Question 3) for the behavior of machine A.



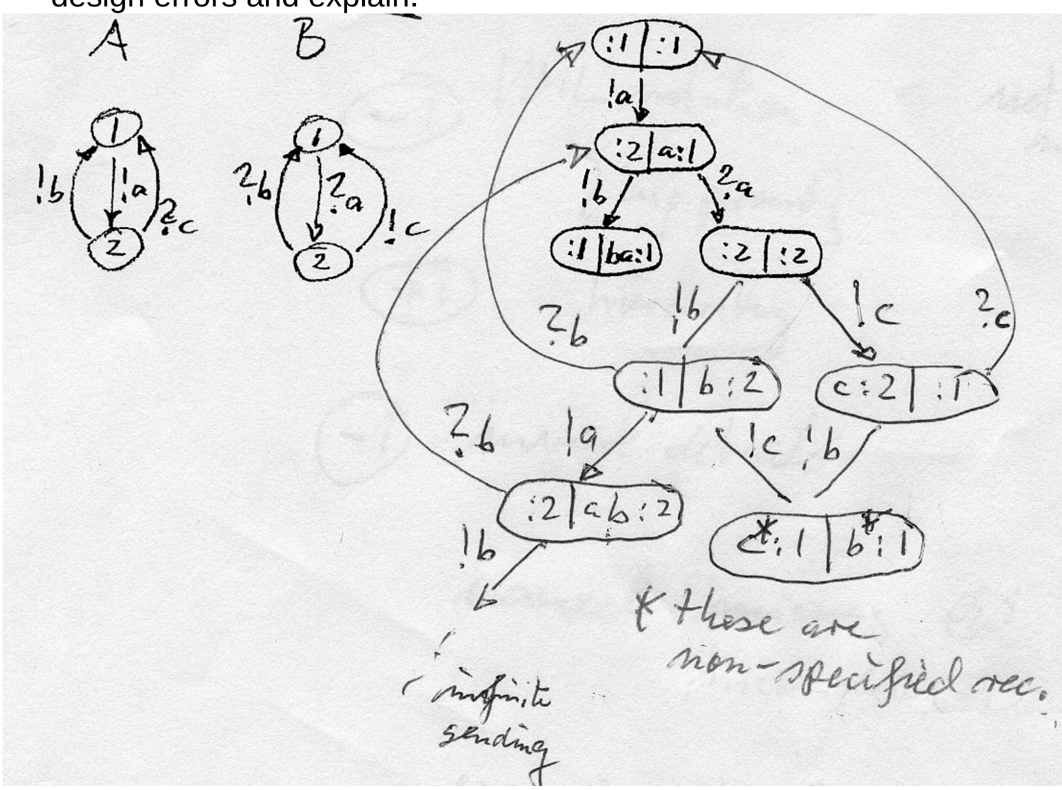


Note: it is not necessary to show the states reached.

(b)



14. **Reachability analysis [6 marks]** We consider two communicating state machines A and B with the behaviors shown in the diagrams below (left side). Please continue the reachability analysis shown by the diagram on the right side and include all interaction sequences that could be generated when in the global state (:2 | :2) machine A may send send b and/or machine B may send c. Indicate if you find any design errors and explain.



Note: It was not asked to explore what can happen from the (:1 | ba :1) global state.

Problem Solving

15. **[10 marks]** A new coffee machine works as follows: The user enters a coin, he then pushes the coffee button, or first pushes the milk button and then pushes the coffee button, after which the machine prepares the coffee (if the milk button was pressed, milk is added) and when ready signals the user that the coffee is ready. The user has to place a cup on the machine before it prepares the coffee. If there is no cup placed when the coffee button is pressed, the machine sends a signal “please place a cup” and waits that a cup is placed. The machine has a sensor that detects whether currently a cup is placed (or not). The machine also has a cancel button; if this button is pushed when a coin was entered and no coffee has been prepared, the coin will be returned and the machine goes back to its initial state.

    (a) Please write down a model of this new coffee machine by

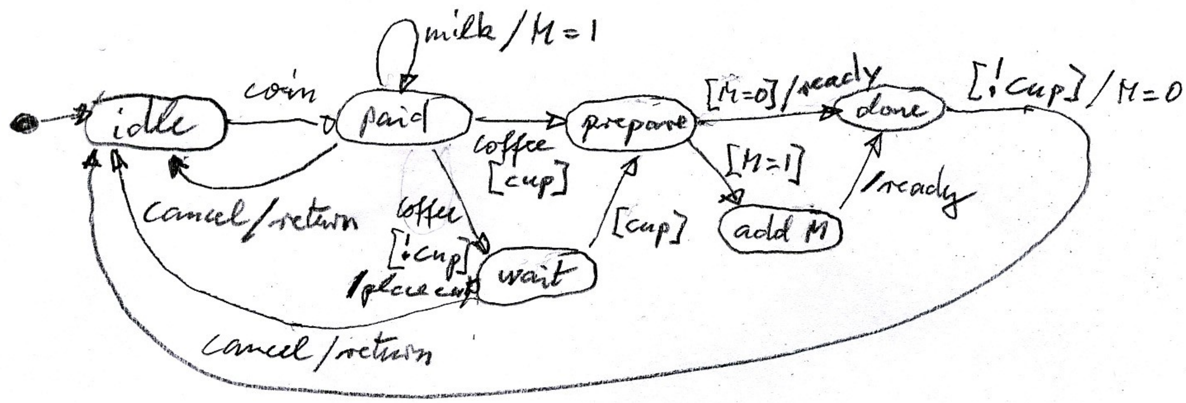
    a. defining its input and output interactions, and
    b. defining its dynamic behavior in the form of an UML state machine, using the standard UML notation used for the example of Question 3.

    (b) Explain what assumptions you have made about the dynamic behavior of the machine – aspects that are not specified in the description above.

(a)

Inputs: coin, coffee, milk, cancel

Outputs: place cup, ready, return



Notes: This model assumes that there is a Boolean variable “cup” that is continuously updated by the sensor. – In this model, all outputs are associated with transitions. It is also possible to have some outputs produced by states (entry or exit actions).

(b) Assumptions (things that are not evident from the description above): (1) The system goes back to its initial state only when the cup of coffee has been taken. (2) The user cannot ask for two milk add-ons.

Annex A : Example of LEX input

Line
number

```
/* File........ parser3.lex
 * Contents.... Exemple of small parser usin LEX
 */

/* compilation:
 *    flex parser3.lex
 *    gcc -o parser3.exe lexyy.c
 */


/* ---------------- Definitions space ----------------- */
%option noyywrap
%{
#include <assert.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

  int students = 0;
%}

DIGIT          [0-9]
STUDENT_ID     [u|s]{DIGIT}{6,7}

/* ------------------- Rules space -------------------- */
%%
{STUDENT_ID}   { ++students; }
\n             { }
.              { }
%%

/* ----------------- User code space ------------------ */
main()
{
  printf("Hit ^Z followed by enter to finish\n");
  yylex();
  printf("--- Students : %d\n", students);

}
```

Annex B : Example of modeling an LTS using the LTSA tool

Specification of example

/** Coffee Machine example - with some problem */

CONTROLLER = (coin[5] -> PAIDFIVE | coin[10] -> PAIDTEN | coinElse -> returnChange -> CONTROLLER),
PAIDFIVE = (coffee -> returnChange -> CONTROLLER),
PAIDTEN = (coffee -> MAKECOFFEE),
MAKECOFFEE = (fillWater -> waterOK -> fillCoffee -> coffeeOK -> heatWater -> warm -> cupOfCoffee -> CONTROLLER).

HARDWARE = (fillWater -> waterOK -> HARDWARE | heatWater -> warm -> HARDWARE | fillCoffee -> PREPARECOFFEE),
PREPARECOFFEE = (coffeeOK -> HARDWARE | tryToFix -> LOOP), LOOP = (tryToFix -> LOOP).

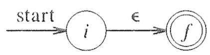
||DETAILEDVIEW = (CONTROLLER || HARDWARE).

Output of the LTSA tool when doing “CHECK” -- “Progress” - note that no deadlock is detected by the tool -

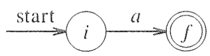
Progress Check...
-- States: 7 Transitions: 10 Memory used: 2761K
Finding trace to cycle...
Finding trace in cycle...
Progress violation: SOME
Trace to terminal set of states:
    coin.10
    coffee
    fillWater
    waterOK
    fillCoffee
    tryToFix
Cycle in terminal set:
    tryToFix
Actions in terminal set:
    tryToFix
Progress Check in: 47ms

Annex C: Algorithm 3.3 (Thompson)

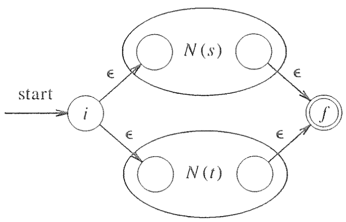
- First parse r into its constituent subexpressions.
- Construct NFA's for each of the basic symbols in r.
  - for ε



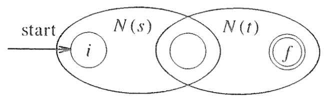
  - for a in Σ



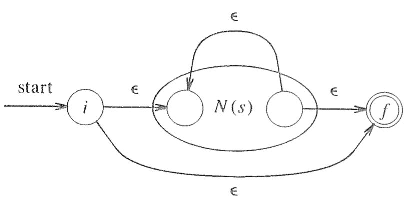
- For the regular expression $s|t$,



- For the regular expression st,



- For the regular expression $s*$,



- For the parenthesized regular expression (s), use $N(s)$ itself as the NFA.

Every time we construct a new state, we give it a distinct name.