Technical University of Denmark

Written examination, May 26, 2020

Course number: 02141

Aids allowed: All written aids are permitted

Changes due to the Covid-19 situation: The exam is conducted as a digital exam taken at home. For this reason answers that can be obtained using a computer need to be properly explained and will not be considered acceptable otherwise. You must submit your answer as a single pdf-file; it is fine to integrate pictures of drawings into the pdf-file, but no attachments to the pdf-file will be considered. If you think there are mistakes in the exam set (where in a written exam you would ask to be contacted by the teacher) please send an e-mail to fnie@dtu.dk for Formal Methods and to albl@dtu.dk for Regular Languages and Context Free Languages. Any announcements resulting from this will be communicated over inside.dtu.dk.

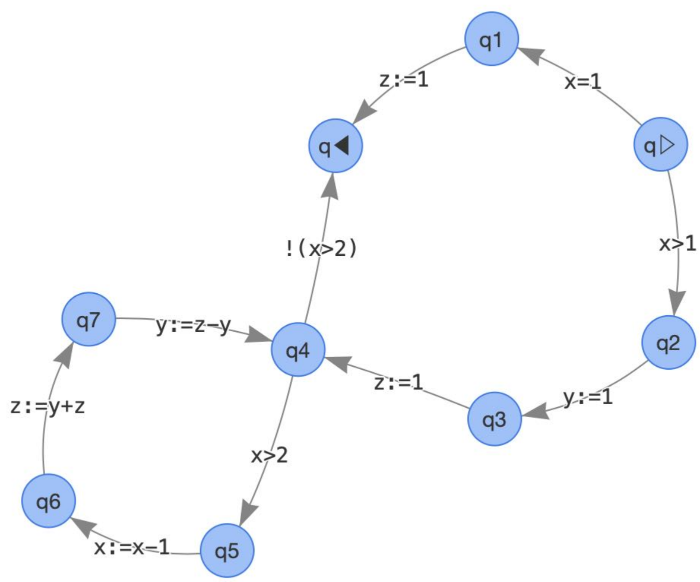
Exam duration: 4 hours

Weighting: 7-step scale

Exercises on Formal Methods

Exercise 1 (12%) Semantics

Consider the following program graph:



Question 1a: *Write a program in Guarded Commands for which* **edges**$(q_\triangleright \rightsquigarrow q_\blacktriangleleft)[\![\cdots]\!]$ *generates the above program graph.*

Consider the complete execution sequence using the semantics in [FM]:

$$\langle q_\triangleright; [\mathtt{x} \mapsto 5, \mathtt{y} \mapsto 4, \mathtt{z} \mapsto 3]\rangle \overset{\omega}{\Longrightarrow}{}^{*} \langle q_\blacktriangleleft; \sigma\rangle$$

Question 1b: *What is σ and how long is ω (i.e. how many steps are taken)? (You do not need to write the entire execution sequence.)*

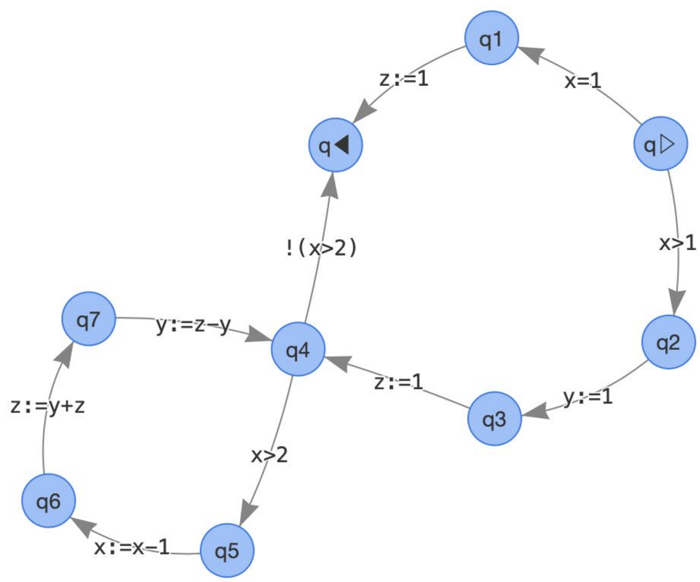
Next we consider the concepts of a program graph being deterministic and/or evolving using the semantics in [FM].

Question 1c: *Is the program graph deterministic? (You should motivate your answer.)*

Question 1d: *Is the program graph evolving? (You should motivate your answer.)*

Exercise 2 (15%) Program Verification and Analysis

Consider the following program graph that is exactly as in the previous exercise:



and consider the following predicate assignment $\mathbf{P}$ where $\mathbf{P}(q_5)$, $\mathbf{P}(q_6)$ and $\mathbf{P}(q_7)$ are missing:

$$
\begin{array}{rcl}
\mathbf{P}(q_{\triangleright}) & = & \mathtt{x} > 0, \mathtt{y} > 0, \mathtt{z} > 0 \\
\mathbf{P}(q_1) & = & \mathtt{x} > 0, \mathtt{y} > 0, \mathtt{z} > 0 \\
\mathbf{P}(q_2) & = & \mathtt{x} > 0, \mathtt{y} > 0, \mathtt{z} > 0 \\
\mathbf{P}(q_3) & = & \mathtt{x} > 0, \mathtt{y} > 0, \mathtt{z} > 0 \\
\mathbf{P}(q_4) & = & \mathtt{x} > 0, \mathtt{y} > 0, \mathtt{z} > 0 \\
\mathbf{P}(q_5) & = & \\
\mathbf{P}(q_6) & = & \\
\mathbf{P}(q_7) & = & \\
\mathbf{P}(q_{\blacktriangleleft}) & = & \mathtt{x} > 0, \mathtt{y} > 0, \mathtt{z} > 0
\end{array}
$$

Question 2a: *Define $\mathbf{P}(q_5)$, $\mathbf{P}(q_6)$ and $\mathbf{P}(q_7)$ such that you get a correct predicate assignment in the sense of [FM].*

Question 2b: *Argue that the predicate assignment is indeed correct in the sense of [FM].*

Moving on to program analysis, it is worth considering whether we could have obtained the same insights using the detection of signs analysis. So consider the following analysis assignment $\mathbf{A}$ where $\mathbf{A}(q_5)$, $\mathbf{A}(q_6)$ and $\mathbf{A}(q_7)$ are missing:

$$
\begin{array}{rcl}
\mathbf{A}(q_\triangleright) & = & \{[\mathtt{x} \mapsto +, \mathtt{y} \mapsto +, \mathtt{z} \mapsto +]\} \\
\mathbf{A}(q_1) & = & \{[\mathtt{x} \mapsto +, \mathtt{y} \mapsto +, \mathtt{z} \mapsto +]\} \\
\mathbf{A}(q_2) & = & \{[\mathtt{x} \mapsto +, \mathtt{y} \mapsto +, \mathtt{z} \mapsto +]\} \\
\mathbf{A}(q_3) & = & \{[\mathtt{x} \mapsto +, \mathtt{y} \mapsto +, \mathtt{z} \mapsto +]\} \\
\mathbf{A}(q_4) & = & \{[\mathtt{x} \mapsto +, \mathtt{y} \mapsto +, \mathtt{z} \mapsto +]\} \\
\mathbf{A}(q_5) & = & \\
\mathbf{A}(q_6) & = & \\
\mathbf{A}(q_7) & = & \\
\mathbf{A}(q_\blacktriangleleft) & = & \{[\mathtt{x} \mapsto +, \mathtt{y} \mapsto +, \mathtt{z} \mapsto +]\}
\end{array}
$$

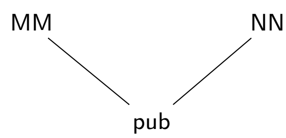
Question 2c: *Is it possible to define $\mathbf{A}(q_5)$, $\mathbf{A}(q_6)$ and $\mathbf{A}(q_7)$ such that you get a semantically correct analysis assignment.? (You should argue for your answer.)*

Exercise 3 (13%) Information Flow

Consider the following program in Guarded Commands

```
i:=0; do i<n -> N[i]:=N[i]+27; i:=i+1 od;
j:=0; do j<m -> M[j]:=M[j]+33; j:=j+1 od
```

together with the following partially ordered set of security properties



Next consider the following possibilities for the security classification $\mathbf{L}_i$:

i	$\mathbf{L}_i(\mathtt{i})$	$\mathbf{L}_i(\mathtt{n})$	$\mathbf{L}_i(\mathtt{N})$	$\mathbf{L}_i(\mathtt{j})$	$\mathbf{L}_i(\mathtt{m})$	$\mathbf{L}_i(\mathtt{M})$
1	pub	pub	NN	pub	pub	MM
2	pub	NN	NN	pub	MM	MM
3	NN	pub	NN	MM	pub	MM
4	NN	NN	NN	MM	MM	MM
5	pub	pub	NN	MM	MM	MM

Question 3a: *For which of the above possibilities for $\mathbf{L}_i$ will the security analysis $\mathbf{sec}[\![\cdots]\!](\{\,\})$ report that there are no explicit or implicit flows that violate $\mathbf{L}_i$? (It is essential that you motivate your answer.)*

Next let us consider the following program in Guarded Commands

```
j:=0; do j<m -> M[j]:=M[j]+33; j:=j+1 od;
i:=0; do i<n -> N[i]:=N[i]+27; i:=i+1 od
```

together with the same partially ordered set of security properties and the same possible security classifications $\mathbf{L}_1$, $\mathbf{L}_2$, $\mathbf{L}_3$, $\mathbf{L}_4$ and $\mathbf{L}_5$.

Question 3b: *Would the answer to the previous question be the same for this program? (It is essential that you motivate your answer.)*

We now consider the following program in Guarded Commands

```
i:=0; j:=0;
do i<n -> N[i]:=N[i]+27; i:=i+1
[] j<m -> M[j]:=M[j]+33; j:=j+1
od
```

together with the same partially ordered set of security properties and the same possible security classifications $\mathbf{L}_1$, $\mathbf{L}_2$, $\mathbf{L}_3$, $\mathbf{L}_4$ and $\mathbf{L}_5$.

Question 3c: *For which of the above possibilities for $\mathbf{L}_i$ will the security analysis $\mathbf{sec}[\![\cdots]\!](\{\,\})$ report that there are no explicit or implicit flows that violate $\mathbf{L}_i$? (It is essential that you motivate your answer.)*

Next let us consider the following program in Guarded Commands

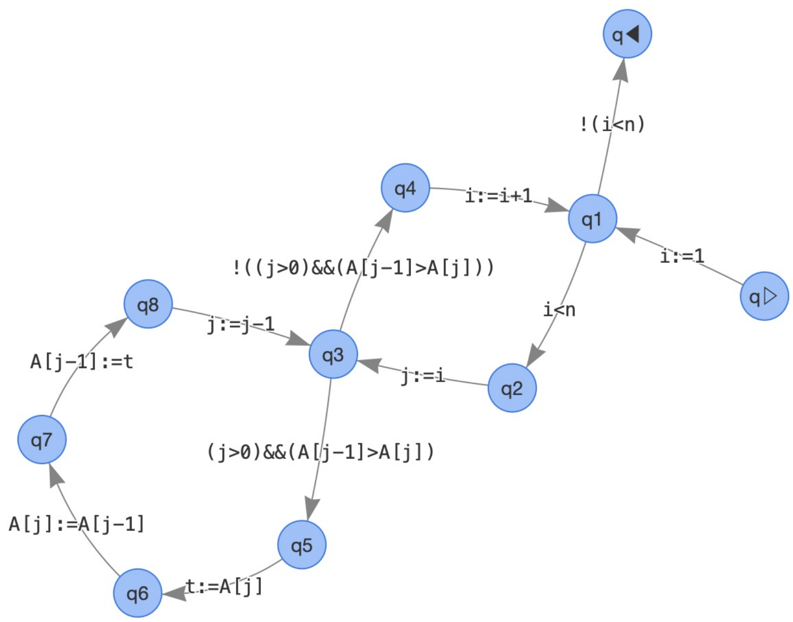
```
j:=0; i:=0;
do j<m -> M[j]:=M[j]+33; j:=j+1
[] i<n -> N[i]:=N[i]+27; i:=i+1
```

together with the same partially ordered set of security properties and the same possible security classifications $\mathbf{L}_1$, $\mathbf{L}_2$, $\mathbf{L}_3$, $\mathbf{L}_4$ and $\mathbf{L}_5$.

Question 3d: *Would the answer to the previous question be the same for this program? (It is essential that you motivate your answer.)*

Exercise 4 (10%) Model Checking

Consider the following program graph (for the version of insertion sort considered in the course):



We next consider a transition system obtained from the program graph (but in a different way compared to [FM] Section 6.4). Informally, just erase all the actions on the edges and take the initial node as the only initial state. More formally, consider a transition system obtained from the program graph such that

- the set of states is the set of nodes in the program graph,
- the set of initial states contains only the state corresponding to the initial node,
- there is a transition from a state to another if and only if there is a directed labelled edge from the former to the latter, i.e. we merely disregard the actions on the edges,
- the set of atomic propositions are the names of the nodes, and
- the labelling function for states therefore trivially maps a state to the set consisting of just that state.

Recall that a state formula Φ of CTL holds on a transition system if and only if it holds on all initial states, which in this case is merely the initial node $q_{\triangleright}$.

Next consider the following formula of CTL:

1 $\mathbf{EF}\, q_{\blacktriangleleft}$

2 $\mathbf{AF}\, q_{\blacktriangleleft}$

3 $\mathbf{AF}\,\mathbf{EF}\, q_{\blacktriangleleft}$

4 $\mathbf{EF}\,\mathbf{AF}\, q_{\blacktriangleleft}$

Question 4a: *Which of the above formulae hold on the transition system? (You should motivate your answer.)*

Next consider the following formula of CTL:

5 $\neg\,\mathbf{E}(\neg\, q_{\triangleright})\mathbf{U}q_{\blacktriangleleft}$

6 $\neg\,\mathbf{E}(\neg\, q_2)\mathbf{U}q_8$

7 $\neg\,\mathbf{E}(\neg\, q_8)\mathbf{U}q_2$

8 $\neg\,\mathbf{E}(\neg\, q_5)\mathbf{U}q_8$

9 $\neg\,\mathbf{E}(\neg\, q_8)\mathbf{U}q_5$

10 $\neg\,\mathbf{E}(\neg\, q_1)\mathbf{U}q_1$

Question 4b: *Which of the above formulae hold on the transition system? (You should motivate your answer.)*

Exercises on Context-free Languages

Exercise 5 (25%)

Function signatures are used in programming languages to specify the type of functions. Consider the following context-free grammar for expressing function signatures in an F#-like style:

F	$\rightarrow$	C	(composed type)
	$\mid$	F`->`F	(function type)
C	$\rightarrow$	B	(basic type)
	$\mid$	C `*` C	(tuple type)
	$\mid$	C `list`	(list type)
	$\mid$	`[`F`]`	(brackets for grouping types)
B	$\rightarrow$	`int`	(integer type)
	$\mid$	`string`	(string type)

the set of non-terminal symbols is $N = \{F, C, B\}$, the set of terminal symbols is $T = \{$`->`, `*`, `list`, `int`, `string`, `[`, `]`$\}$ and the initial symbol is F. Function signatures are all strings accepted by the grammar.

(a) Design a set of datatypes that are suitable to store abstract representations (ASTs) of function signatures and how the function signature

`[ int list * string ] -> string`

would be represented as a value of your datatype.

(b) Provide the precedence rules of the type composition operators `->`, `*` and `list` by filling out the following table. In cell (x, y) write "yes" if, according to the above grammar, x has precedence over y (i.e. x binds more tightly than y), and write "no" otherwise.

	`->`	`*`	`list`
`->`	X		
`*`		X	
`list`			X

(c) If the answer to exercise (b) includes two operators whose precedence is undefined according to the above grammar (they don't have precedence over each other), provide an example of a function signature that has two distinct parse trees, each encoding a different precedence. Otherwise, briefly explain why it is not possible to find such an expression.

(d) In F# operator `->` is right-associative. Is it also like this in the provided grammar? If the answer is yes, provide a brief explanation. If the answer is no, explain why and provide a new grammar that enforces associativity of `->` as in F# and that accepts the same language of the original grammar.

(e) We focus now on the subset of function signatures given by the regular expression $(\texttt{int} + \texttt{string})(\texttt{->}(\texttt{int} + \texttt{string}))^*$. Provide a context-free grammar that accepts the same language as the regular expression.

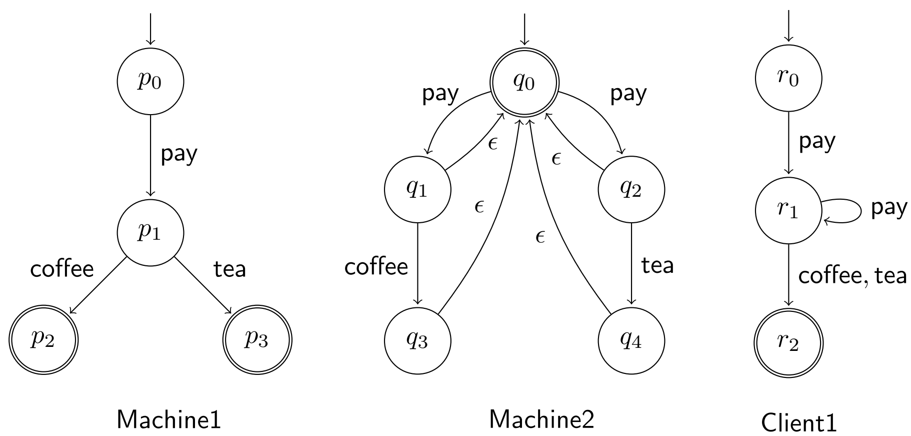
(f) Construct a Pushdown Automaton (PDA) that accepts the same language as the grammar in exercise (e). Use the construction that we saw in class (and described in book [HMU] 02141: Automata Theory and Languages - edited by Hanne Riis Nielson) to translate a context-free grammar into an equivalent, non-deterministic PDA that accepts by empty stack.

(g) The PDA obtained in exercise (f) is non-deterministic. Can you build a deterministic PDA instead? If no, explain why. If yes, provide the deterministic PDA.

Exercises on Regular Languages

Exercise 6 (25%)

In this exercise we consider models of coffee machines and their clients. Coffee machines Machine1, Machine2 and client Client1 are modelled with finite state automata given as transition diagrams below:



Client Client2 instead is modelled with the regular expression

$$\mathsf{pay}^*(\mathsf{coffee} + \mathsf{tea})$$

(a) For each client $C \in \{\mathsf{Client1}, \mathsf{Client2}\}$ and each coffee machine $M \in \{\mathsf{Machine1}, \mathsf{Machine2}\}$ answer the following question: Does client C accept words that are *not* accepted by machine M? Respond by

- providing a table, like the one below, filled with either "yes" or "no", and
- providing a justification for each answer: if the answer is "yes", provide one example (as short as possible), if the answer is "no" provide a very brief explanation.

	Machine1	Machine2
Client1		
Client2		

(b) We want now to make clients interact with machines and for that purpose we define a composition operator $\triangleright$ as follows.

Let $A = \langle Q_A, \Sigma, \delta_A, q_0^A, F_0^A \rangle$ and $B = \langle Q_B, \Sigma, \delta_B, q_0^B, F_0^B \rangle$ be ϵ-NFA. We define the *composition* of A and B, denoted $A \triangleright B$, as the ϵ-NFA $\langle Q, \Sigma, \delta, q_0, F_0 \rangle$ such that

- $Q = (Q^A \times Q^B) \cup \{\mathsf{error}\}$;

- $\delta : Q \times \Sigma \to P(Q)$ is such that

$$\delta((s_A, s_B), a) = \begin{cases} \left(\begin{array}{l} \{(s'_A, s_B) \mid s'_A \in \delta_A(s_A, a)\} \cup \\ \{(s_A, s'_B) \mid s'_B \in \delta_B(s_B, a)\} \end{array} \right) & \text{if } a = \epsilon \\ \{\mathsf{error}\} & \text{if } a \in (\Sigma \setminus \{\epsilon\}) \text{ , } \delta_A(s_A, a) \neq \emptyset \\ & \text{ and } \delta_B(s_B, a) = \emptyset \\ \delta_A(s_A, a) \times \delta_B(s_B, a) & \text{otherwise} \end{cases}$$

- $q_0 = (q_0^A, q_0^B)$;
- $F = F_0^A \times F_0^B$.

where we assume that state error is a new state neither contained in Q^A nor in Q^B.

Construct the automaton $\mathsf{Client1} \triangleright \mathsf{Machine1}$ and provide it as a transition diagram. Depict only the reachable states.

(c) Construct the automaton $\mathsf{Client1} \triangleright \mathsf{Machine2}$ and provide it as a transition diagram.

(d) Convert $\mathsf{Client2}$ into an ϵ-NFA using the procedure seen in the course. Provide the result as a transition diagram and do not apply any optimisation (i.e. do not remove ϵ-transitions).

(e) Convert $\mathsf{Machine2}$ into a DFA using the procedure seen in the course. Provide the result as a transition diagram. Do not include unreachable states. Do include the state $\emptyset$ if it is reachable.