Concurrency Theory WS 2013/2014
Chair for Software Modeling and Verification
Rheinisch-Westfälische Technische Hochschule Aachen

Prof. Dr. Ir. Joost-Pieter Katoen
apl. Prof. Dr. Thomas Noll
S. Chakraborty, B. Kaminski, H. Wu

Concurrency Theory WS 2013/2014

— 2nd Exam —

First Name: _____

Second Name: _____

Matriculation Number: _____

Degree Programme (please mark):

- ◦ CS Bachelor
- ◦ CS Master
- ◦ CS Lehramt
- ◦ SSE Master
- ◦ Other: _____

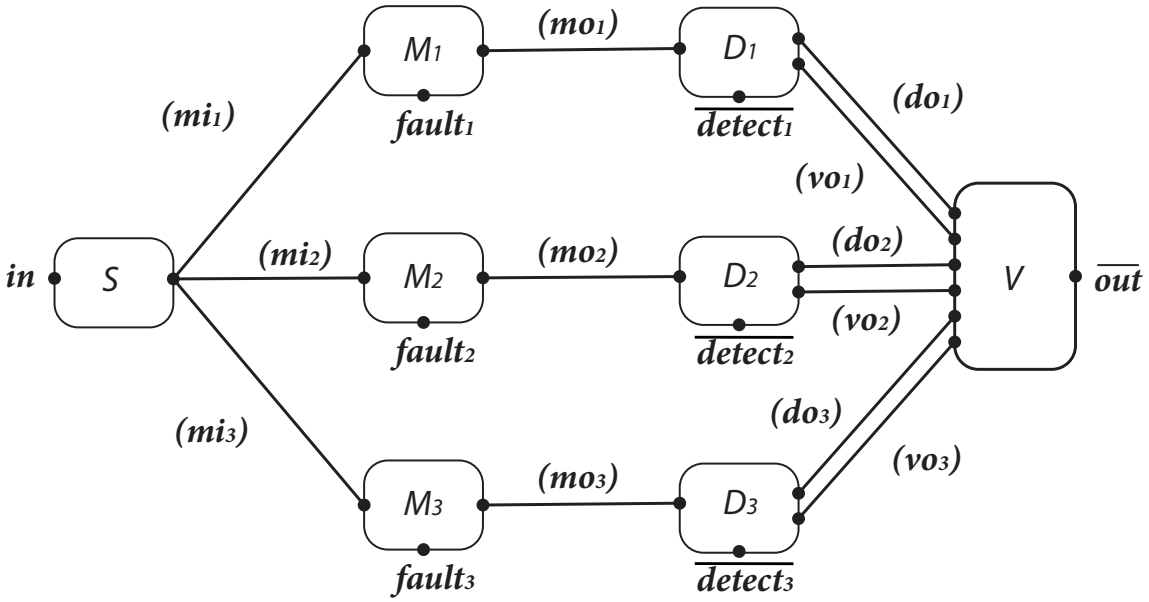
General Information:

- Mark every sheet with your **matriculation number**.
- Check that your copy of the exam consists of **12 sheets (24 pages)**.
- Duration of exam: **120 minutes**.
- No helping materials (e.g. books, notes, slides) are permitted.
- Give your solution on the respective sheet. Also use the backside if necessary. If you need more paper, ask the assistants.
- Write with blue or black ink; do **not** use a pencil or red ink.
- Make sure all electronic devices are switched off and are nowhere near you.
- Any attempt at deception leads to failure for this exam, even if it is detected only later.

	Σ Points	Points obtained
Task 1	**16**	
Task 2	**27**	
Task 3	**22**	
Task 4	**21**	
Task 5	**23**	
Task 6	**11**	
Σ	**120**	

Task 1 (Modeling with Value–Passing CCS) (11+5 Points)

In this task, you are requested to model a technique called Triple-Modular Redundancy (TMR) with error detection for fault tolerance in distributed systems in value-passing CCS. The overview of the system is provided in following diagram. In TMR, to increase the reliability three copies of system (called components) are used. Three input value copies are sent to each component by using a splitter (S) respectively; the voter then accepts the results from each of the component and outputs the majority value. In our case, one component consists of a module (M_i) and a detector (D_i) ($1 \leq i \leq 3$). A functional module normally will get and pass the same value to D_i (via mo_i) as from what it gets, but if a fault occurs, differing values will be passed. The detector will first record the value and pass the value to voter (via do_i), then later compare this value with the feedback value from the voter (via vo_i) to determine a fault situation. Here, we assume the set Val of input values is given by $Val = \{0, 1\}$, and *only one* of the three modular can be faulty and the splitter, voter and detectors work always errorless.
Additional information: In the diagram, the action name in ( ) means it is an internal synchronization action, otherwise it is a external action.



(a) Model this system using value-passing CCS! You should reuse the component definition (with index) when it is possible.

(b) Now we add a new operator called *hide* with respect to an action set H in CCS language, which is defined by following SOS rule:

$$(\text{hide}_1)\ \frac{P \xrightarrow{a} P'}{hide\ H\ P \xrightarrow{a} hide\ H\ P'}\ (a \notin H \cup \overline{H})$$

$$(\text{hide}_2)\ \frac{P \xrightarrow{a} P'}{hide\ H\ P \xrightarrow{\tau} hide\ H\ P'}\ (a \in H \cup \overline{H})$$

Informally speaking, this operator can turn all the actions in H into τ.

Let $Cycler_1 = fault_1.\overline{detect_1}.Cycler_1$ and let TMR be your modelled system. Argue why your system satisfies the verification condition that

$$hide\ \{in, out, fault_2, detect_2, fault_3, detect_3\}\ TMR \approx Cycler_1$$

which states that if a fault occurs in M_1, D_1 should always detect this fault.

Task 2 (Labeled Transition Systems) (14+7+6 Points)

(a) Consider the following CCS process definition:

$$\begin{aligned} A &= \big((B \parallel C) + D\big) \setminus \{\mathrm{com}\} \\ B &= \mathrm{a.com}.B + \mathrm{b}.nil \\ C &= D + E \\ D &= \overline{\mathrm{com}}.D \\ E &= \mathrm{b}.C \end{aligned}$$

Derive all legal outgoing transitions $A \xrightarrow{\alpha} A'$ by giving their derivation tree!

(b) Reconsider the CCS process definition from Task 2 (a):

$$\begin{aligned} A &= \big((B \parallel C) + D\big) \setminus \{\text{com}\} \\ B &= \text{a.com}.B + \text{b}.nil \\ C &= D + E \\ D &= \overline{\text{com}}.D \\ E &= \text{b}.C \end{aligned}$$

Draw LTS(A) and label the nodes with the corresponding CCS processes!

(c) Give the trace language $\mathrm{Tr}\big((B \parallel D) \setminus \{\mathrm{com}\}\big)$ of process $(B \parallel D) \setminus \{\mathrm{com}\}$!

Task 3 (HML and Bisimulation) (15+7 Points)

Given are the following three CCS processes:

$$
\begin{array}{lll}
A = b.B + a.C + a.d.C & D = a.d.E + a.F + b.G + a.I & J = a.L + b.K + a.M \\
B = a.A + a.d.C & E = b.H + c.d.E & K = a.O + a.N \\
C = b.A + c.d.C & F = b.D + c.d.F & L = d.M \\
 & G = a.d.E + a.H & M = b.J + c.N \\
 & H = b.G + a.d.F + a.E & N = d.M \\
 & I = c.d.E + b.D & O = a.L + a.M
\end{array}
$$

(a) Draw the LTSs for A, D, and J respectively. Prove or disprove $A \sim D$, $A \sim J$ and $D \sim J$. For proving or disproving that two processes are strongly bisimilar, you can use the game characterization of bisimilarity. For disproving you may provide an HML formula which is satisfied by one process but not by the other.

(b) Express the property that actions a and b occur in alternation in HML+Recursion! Note that this does *not* imply that a and b are *strictly* followed by each other (i.e. between any two occurrences of a a b occurs and vice versa, but between an a and a b the other symbols c and d may occur). Check whether process A satisfies the property or not!

Task 4 (Preservation of Strong Bisimilarity) (6+15 Points)

(a) Let $\text{synhide}_L(\,\cdot\,)$, where L is a set of actions, be a unary CCS operator with the following semantics:

$$\text{(synhide)}\ \frac{P \xrightarrow{\alpha} P' \qquad \forall \beta \in L\colon P \neq \beta.P'}{\text{synhide}_L(P) \xrightarrow{\alpha} \text{synhide}_L(P')}$$

Prove or disprove: $\text{synhide}_L(\cdot)$ preserves strong bisimilarity, i.e. for any two strongly bisimilar processes $S \sim T$ it holds that $\text{synhide}_L(S) \sim \text{synhide}_L(T)$.

(b) Let † be a binary CCS operator with the following semantics:

$$\text{(dag1)}\ \frac{P \overset{\alpha}{\not\longrightarrow} \qquad Q \xrightarrow{\alpha} Q'}{P \dagger Q \xrightarrow{\alpha} Q'}$$

$$\text{(dag2)}\ \frac{P \xrightarrow{\alpha} P' \qquad Q \overset{\alpha}{\not\longrightarrow}}{P \dagger Q \xrightarrow{\alpha} P'}$$

$$\text{(dag3)}\ \frac{P \xrightarrow{\alpha} P' \qquad Q \xrightarrow{\alpha} Q'}{P \dagger Q \xrightarrow{\alpha} P' \dagger Q'}$$

Prove or disprove: † preserves strong bisimilarity, i.e. for any two strongly bisimilar processes $S \sim T$ and any other process R it holds that $S \dagger R$ is strongly bisimilar to $T \dagger R$ (and you may omit in your proof the analogous case for $R \dagger S$ is strongly bisimilar to $R \dagger T$).

Task 5 (From Modified CCS to Petri Nets) (8+8+7 Points)

In the lecture, we have introduce an occurrence net (Petri net) semantics for CCS processes. Now we modify the parallel composition operation $\|$ to a new operator ($\|_A$) w.r.t to an action set A. Informally speaking, this operator requests a forced synchronization on an action between two processes if the action belongs to the set A. The formal SOS rules for this operator are as follows:

$$\text{(Syn)}\ \frac{P \xrightarrow{a} P' \qquad Q \xrightarrow{a} Q' \qquad a \in A}{P \parallel_A Q \xrightarrow{a} P' \parallel_A Q'}$$

$$\text{(Par}_1\text{)}\ \frac{P \xrightarrow{a} P' \qquad a \notin A}{P \parallel_A Q \xrightarrow{a} P' \parallel_A Q} \qquad \text{(Par}_2\text{)}\ \frac{Q \xrightarrow{a} Q' \qquad a \notin A}{P \parallel_A Q \xrightarrow{a} P \parallel_A Q'}\ (a \notin A)$$

(a) Give the occurrence net semantics for the process P!

$$\begin{aligned} P &= x.a.P + y.a.Q \\ Q &= z.a.Q \end{aligned}$$

(b) Formally define the occurrence net semantics for the operator $\|_A$ and give the occurrence net semantics for $P \|_{\{a\}} Q$ based on your definition! Note that, since the occurrence net is infinite, you should stop as soon as some part in the occurrence net repeats.

(c) Compute the McMillan prefix of the resulting occurrence net from **(b)**.

Task 6 (Petri net Acceptable Languages) (11 Points)

Let Σ be a finite alphabet and let $N = (P, T, F, M_0, \lambda)$ be a labelled Petri net in which all transitions in T are labelled by a labeling function $\lambda\colon T \to \Sigma$. Then the trace language $\mathrm{Tr}(N)$ of N is defined as the following set:

$$\left\{ w = \lambda(a_1)\cdots\lambda(a_k) \;\middle|\; M_0 \xrightarrow{a_1} M_1 \xrightarrow{a_1} \cdots \xrightarrow{a_k} M_k \text{ is a complete sequential run of } N \right\}.$$

A language $L \subseteq \Sigma$ is called Petri net recognizable, if there exists a labelled Petri net N such that $\mathrm{Tr}(N) = L$.

Provide an exact description of the language that is recognized by the following Petri net! Your description shall not make any reference back to the Petri net itself!