Skriftlig Eksamen

DM861 Concurrency Theory

Institut for Matematik og Datalogi, Syddansk Universitet, Odense

06/01/2020

Instructions

Modality The following conventions are used/suggested for answers in pure text (ASCII) form:

- Subscripts are numerical and can be rendered as numbers (e.g. `s0` for s_0) or using latex notation (e.g. `s_0` for s_0).
- Superscripts are limited to primes (e.g. `s'`) and seconds (e.g. `s''`).
- The arrow $\longrightarrow$ is `--->`; the labelled arrow is `-a->`; the mapping arrow $\mapsto$ is `|-->`.
- Often common symbols are rendered by the name of their (standard) latex macros, e.g. σ is `\sigma`, Σ is `\Sigma`.
- Expression evaluation judgements ($\Sigma(\mathsf{p}) \vdash e \downarrow v$) can be written `\Sigma(p) |- e \eval v`.
- $\oplus$ is `\oplus` or `+` when clear from the context.
- $\cup$ is `\cup` or `U` when clear from the context.

Feel free to use your own notation provided it is properly introduced (in a dedicated legend or where it is first used). Mathematical operators and symbols in Unicode are also fine.

Supplementary material You may only consult the material in the Appendix. EPP is an acronym for EndPoint Projection.

$$\frac{}{\mathsf{nat}(\mathsf{z})}\ [\textsc{Nat-Z}] \qquad \frac{\mathsf{nat}(X)}{\mathsf{nat}(\mathsf{s}X)}\ [\textsc{Nat-S}] \qquad \frac{\mathsf{nat}(Y)}{\mathsf{add}(\mathsf{z},Y,Y)}\ [\textsc{Add-Z}] \qquad \frac{\mathsf{add}(X,Y,Z)}{\mathsf{add}(\mathsf{s}X,Y,\mathsf{s}Z)}\ [\textsc{Add-S}]$$

System 1: Peano Arithmetic.

Question 1

Consider the rules in System 1, where X, Y, and Z are schematic variables. Some intuition about the rules: z represents the number zero; for any X, $\mathsf{nat}(X)$ means "X is a natural number"; for any X, $\mathsf{s}X$ means "the successor of X"; for any X, Y, and Z, $\mathsf{add}(X,Y,Z)$ means "Z is the result of adding Y to X".

Answer the following questions and motivate your answers formally.

(a) Are $\mathsf{nat}(\mathsf{sz})$, $\mathsf{nat}(\mathsf{ss})$, $\mathsf{add}(\mathsf{sz},\mathsf{z},\mathsf{sz})$, and $\mathsf{add}(\mathsf{z},\mathsf{s},\mathsf{sz})$ derivable using the rules in System 1?

(b) Is rule [Add-S2] below derivable in System 1?

$$\frac{\mathsf{add}(X,Y,Z)}{\mathsf{add}(\mathsf{ss}X,Y,\mathsf{ss}Z)}\ [\textsc{Add-S2}]$$

(c) Is rule [Add-S3] below derivable in System 1?

$$\frac{\mathsf{add}(X,\mathsf{s}Y,Z)}{\mathsf{add}(\mathsf{s}X,Y,Z)}\ [\textsc{Add-S3}]$$

(d) Is rule [Add-S3] above admissible in System 1?

(e) Is rule [Add-Comm] below admissible in System 1?

$$\frac{\mathsf{add}(Y,X,Z)}{\mathsf{add}(X,Y,Z)}\ [\textsc{Add-Comm}]$$

Question 2

(a) Write a projectable choreography in the language of Recursive Choreographies (see Appendix) for:
- p sends a product name to q;
- q asks r whether the product name is in stock;
- r replies to q with the quantity of product that is available;
- p sends the quantity of product it wishes to buy to q;
- q decides:
  - if the desired quantity is available:
    - q confirms the order to p;
    - p sends the shipping address to r.
  - if the desired quantity is not available:

    - nothing happens.

  You are free to insert selections as necessary to make the choreography projectable.

  You do not need to define the functions used by processes in interactions, and you do not have to provide a proof that the choreography is projectable.

(b) Extend the choreography you wrote for (a) such that, if q decides that the desired quantity is not available (second-last item), instead of "nothing happens" the following takes place:

- q sends p the available quantity
- p decides
  - if the available quantity is at least 80% of the desired one:
    - p sends the shipping address to r.
    - q sends a voucher to p;
  - otherwise
    - nothing happens

Question 3

Show all reduction chains for the configuration $\langle C, \Sigma, \emptyset \rangle$ in the language of Recursive Choreographies (see Appendix), where C is

```
(if p.e then
  q.1 -> p.x1;
  p -> q.ok;
  p.2 -> q.x2;
  0
else
  q.1 -> p.x2;
  p -> q.ko;
  q.2 -> p.x2;
  0);0
```

and Σ is such that $\mathsf{e}(\Sigma(\mathsf{p})) = \mathtt{true}$.

Question 4

Let C be the choreography `p -> q; r -> s; p -> s; 0` in the language of Simple Concurrent Choreographies (see Appendix).

(a) Write the EPP of C.

(b) Show that C is operationally correspondent to its EPP, by matching all the possible transitions by C and its (multi-step) derivatives with those of its EPP and vice versa.

Question 5

Consider the choreography and network below, in the setting of Simple Concurrent Choreographies.

```
q -> p; p -> s; s -> q; r -> p; 0
```

```
p[q?;s!;r?;0]  |  q[p!;s!;0]  |  r[p!;0]  |  s[p?;q!;0]
```

(a) Are they operationally correspondent (that is, do their transitions mimic each other, see also previous question)? Motivate your answer informally.

(b) Give a formal argument for your answer to the previous question. (Hint: use multi-step derivatives/transitions.)

Question 6

Write the EPP of the following choreography, given in the language for Recursive Choreographies (see Appendix).

```
p.prod -> q.x;
q.price(x) -> p.y;
q.discount(x) -> p.z;
(if p.(y - z < 100) then
  p -> b.ok;
  p -> q.ok;
  p.money(y-z) -> b.m;
  q.prod(x) -> p.res;
  0
else
  p -> b.ko;
  p -> q.ko;
  0
);0
```

Question 7

Consider the set of choreographic procedures $\mathscr{C}$ that consists only of the following procedure definition, given in the language for Recursive Choreographies (see Appendix).

```
Sell(p,q,b) =
  p.prod -> q.x;
  q.price(x) -> p.y;
  q.discount(x) -> p.z;
  (if p.(y - z < 100) then
    p -> b.ok;
    p -> q.ok;
    p.money(y-z) -> b.m;
    q.prod(x) -> p.res;
    0
  else
    p -> b.ko;
    p -> q.ko;
    q.prices := changePrices;
    Sell(p,q,b);
    0
  );0
```

Write the EPP of $\mathscr{C}$, that is, the set of procedure definitions $\mathscr{P}$ that results by projecting $\mathscr{C}$.

Question 8

(a) A network N in the language of Simple Concurrent Processes (see Appendix) *can terminate* if it is either the terminated network $\mathbf{0}$ (where all processes have no code to execute) or there exists a sequence of labels $\vec{\mu}$ such that $N \xrightarrow{\vec{\mu}} \mathbf{0}$.
Can the network below terminate? Answer yes or no.

```
p[r?;t!;0] | q[t?;0] | r[p!;0] | s[t!;0]  | t[s?;q!;0]
```

(b)
- If it can, provide a choreography in the language of Simple Concurrent Choreographies (see Appendix) whose EPP is the network above.
- If it cannot, prove that there is no sequence of labels $\vec{\mu}$ such that $N \xrightarrow{\vec{\mu}} \mathbf{0}$.

Question 9

(a) A network N in the language of Recursive Processes (see Appendix) *can terminate under* $\mathscr{P}$ if it is either the terminated network $\mathbf{0}$ (where all processes have no code to execute) or there exist Σ, a sequence of labels $\vec{\mu}$ and Σ' such that $\langle N, \Sigma, \mathscr{P}\rangle \xrightarrow{\vec{\mu}} \langle \mathbf{0}, \Sigma', \mathscr{P}\rangle$.
Can this network terminate under $\emptyset$ (the empty $\mathscr{P}$)? (See the previous question for the definition of termination.) Answer yes or no.

```
p[q?z; r!z;0] |
q[p!z; r&{ ok: r?x;0, ko: s?y;0 }];0 |
r[p?w; (if w then (q⊕ok;s⊕ok;q!w;0) else (q⊕ko;s⊕ko;0));0] |
s[r&{ ok: 0, ko: q!y;0 };0]
```

(b)
- If it can, provide a choreography in the language of Recursive Choreographies (see Appendix) whose EPP is the network above.
- If it cannot, prove that there is no Σ, $\vec{\mu}$ and Σ' such that $\langle N, \Sigma, \emptyset\rangle \xrightarrow{\vec{\mu}} \langle \mathbf{0}, \Sigma', \emptyset\rangle$.

Simple Concurrent Choreographies

$$C ::= \mathsf{p} \text{ -> } \mathsf{q}; C \mid \mathbf{0}$$

$$\frac{}{\mathsf{p} \text{ -> } \mathsf{q}; C \xrightarrow{\mathsf{p\text{->}q}} C}\ [\textsc{Com}] \qquad \frac{C \xrightarrow{\mu} C' \quad \{\mathsf{p},\mathsf{q}\} \# \mathsf{pn}(\mu)}{\mathsf{p} \text{ -> } \mathsf{q}; C \xrightarrow{\mu} \mathsf{p} \text{ -> } \mathsf{q}; C'}\ [\textsc{Delay}]$$

$$\mathsf{pn}(\mathbf{0}) = \emptyset$$
$$\mathsf{pn}(\mathsf{p} \text{ -> } \mathsf{q}; C) = \mathsf{pn}\left(\mathsf{p} \text{ -> } \mathsf{q}\right) \cup \mathsf{pn}(C)$$

$$\mathsf{pn}\left(\mathsf{p} \text{ -> } \mathsf{q}\right) \ = \ \{\mathsf{p},\mathsf{q}\}$$

Simple Concurrent Processes

$$P, Q, R ::= \mathsf{p}!; P \mid \mathsf{p}?; P \mid \mathbf{0}$$

$$\frac{}{\mathsf{p}[\mathsf{q}!; P] \mid \mathsf{q}[\mathsf{p}?; Q] \xrightarrow{\mathsf{p\text{->}q}} \mathsf{p}[P] \mid \mathsf{q}[Q]}\ [\textsc{Com}] \qquad \frac{N \xrightarrow{\mu} N'}{N \mid M \xrightarrow{\mu} N' \mid M}\ [\textsc{Par}]$$

Recursive Choreographies

$$\begin{aligned} C &::= I; C \mid \mathbf{0} \\ I &::= \mathsf{p}.e \text{ -> } \mathsf{q}.x \mid \mathsf{p} \text{ -> } \mathsf{q}.\textsc{l} \mid \mathsf{p}.x \mathrel{\mathbf{:=}} e \mid \mathsf{if}\ \mathsf{p}.e\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2 \mid X(\vec{\mathsf{p}}) \\ \mathscr{C} &::= X(\vec{\mathsf{p}}) = C, \mathscr{C} \mid \emptyset \end{aligned}$$

$$\frac{\Sigma(\mathsf{p}) \vdash e \downarrow v}{\langle \mathsf{p}.x \mathrel{\mathbf{:=}} e; C, \Sigma, \mathscr{C} \rangle \xrightarrow{\tau @ \mathsf{p}} \langle C, \Sigma\left[\mathsf{p}.x \mapsto v\right], \mathscr{C} \rangle}\ [\textsc{Local}]$$

$$\frac{\Sigma(\mathsf{p}) \vdash e \downarrow v}{\langle \mathsf{p}.e \text{ -> } \mathsf{q}.x; C, \Sigma, \mathscr{C} \rangle \xrightarrow{\mathsf{p}.v\text{->}\mathsf{q}} \langle C, \Sigma\left[\mathsf{q}.x \mapsto v\right], \mathscr{C} \rangle}\ [\textsc{Com}]$$

$$\frac{}{\langle \mathsf{p} \text{ -> } \mathsf{q}.\textsc{l}; C, \Sigma, \mathscr{C} \rangle \xrightarrow{\mathsf{p} \text{ -> } \mathsf{q}.\textsc{l}} \langle C, \Sigma, \mathscr{C} \rangle}\ [\textsc{Sel}]$$

$$\frac{i = 1 \text{ if } \Sigma(\mathsf{p}) \vdash e \downarrow \mathtt{true},\ i = 2 \text{ otherwise}}{\langle \mathsf{if}\ \mathsf{p}.e\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2; C, \Sigma, \mathscr{C} \rangle \xrightarrow{\tau @ \mathsf{p}} \langle C_i \mathbin{⨾} C, \Sigma, \mathscr{C} \rangle}\ [\textsc{Cond}]$$

$$\frac{X(\vec{\mathsf{q}}) = C \ \in \ \mathscr{C} \qquad \mathsf{r} \in \vec{\mathsf{p}}}{\langle X(\vec{\mathsf{p}}); C', \Sigma, \mathscr{C}\rangle \xrightarrow{\tau @ \mathsf{r}} \langle\langle \vec{\mathsf{p}} \setminus \mathsf{r}; C'\rangle X(\vec{\mathsf{p}}); C[\vec{\mathsf{p}}/\vec{\mathsf{q}}] \fatsemi C', \Sigma, \mathscr{C}\rangle} \ [\textsc{CallFirst}]$$

$$\frac{\mathsf{r} \in \vec{\mathsf{q}} \qquad \vec{\mathsf{q}} \setminus \mathsf{r} \text{ nonempty}}{\langle\langle \vec{\mathsf{q}}; C'\rangle X(\vec{\mathsf{p}}); C, \Sigma, \mathscr{C}\rangle \xrightarrow{\tau @ \mathsf{r}} \langle\langle \vec{\mathsf{q}} \setminus \mathsf{r}; C'\rangle X(\vec{\mathsf{p}}); C, \Sigma, \mathscr{C}\rangle} \ [\textsc{CallEnter}]$$

$$\frac{}{\langle\langle \mathsf{q}; C'\rangle X(\vec{\mathsf{p}}); C, \Sigma, \mathscr{C}\rangle \xrightarrow{\tau @ \mathsf{q}} \langle C, \Sigma, \mathscr{C}\rangle} \ [\textsc{CallDone}]$$

$$\frac{\langle C, \Sigma, \mathscr{C}\rangle \xrightarrow{\mu} \langle C', \Sigma', \mathscr{C}\rangle \qquad \mathsf{pn}(I) \# \mathsf{pn}(\mu)}{\langle I; C, \Sigma, \mathscr{C}\rangle \xrightarrow{\mu} \langle I; C', \Sigma', \mathscr{C}\rangle} \ [\textsc{Delay}]$$

$$\frac{\langle C_1, \Sigma, \mathscr{C}\rangle \xrightarrow{\mu} \langle C_1', \Sigma', \mathscr{C}\rangle \qquad \langle C_2, \Sigma, \mathscr{C}\rangle \xrightarrow{\mu} \langle C_2', \Sigma', \mathscr{C}\rangle \qquad \mathsf{p} \notin \mathsf{pn}(\mu)}{\langle \mathsf{if}\ \mathsf{p}.e\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2; C, \Sigma, \mathscr{C}\rangle \xrightarrow{\mu} \langle \mathsf{if}\ \mathsf{p}.e\ \mathsf{then}\ C_1'\ \mathsf{else}\ C_2'; C, \Sigma', \mathscr{C}\rangle} \ [\textsc{DelayCond}]$$

$$\begin{aligned}
\mathsf{pn}\,(I; C) &= \mathsf{pn}(I) \cup \mathsf{pn}(C) \\
\mathsf{pn}(\mathbf{0}) &= \emptyset \\
\mathsf{pn}\,(\mathsf{p}.e \texttt{->} \mathsf{q}.x) &= \{\mathsf{p}, \mathsf{q}\} \\
\mathsf{pn}\,(\mathsf{p} \texttt{->} \mathsf{q}.\textsc{l}) &= \{\mathsf{p}, \mathsf{q}\} \\
\mathsf{pn}\,(\mathsf{p}.x \ \texttt{:=}\ e) &= \{\mathsf{p}\} \\
\mathsf{pn}\,(\mathsf{if}\ \mathsf{p}.e\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2) &= \{\mathsf{p}\} \cup \mathsf{pn}(C_1) \cup \mathsf{pn}(C_2) \\
\mathsf{pn}(X(\vec{\mathsf{p}})) &= \ \{\vec{\mathsf{p}}\}\ .
\end{aligned}$$

Recursive Processes

$$\begin{aligned}
P &::= I; P \mid \mathbf{0} \\
I &::= \mathsf{p}!e \mid \mathsf{p}?x \mid \mathsf{p} \oplus \textsc{l} \mid \mathsf{p} \,\&\{\textsc{l}_i : P_i\}_{i \in I} \mid x \ \texttt{:=}\ e \mid \mathsf{if}\ e\ \mathsf{then}\ P\ \mathsf{else}\ Q \mid X(\vec{\mathsf{p}}) \\
\mathscr{P} &::= X(\vec{\mathsf{p}}) = P, \mathscr{P} \mid \emptyset
\end{aligned}$$

$$\frac{\Sigma(\mathsf{p}) \vdash e \downarrow v}{\langle \mathsf{p}[\mathsf{q}!e; P] \mid \mathsf{q}[\mathsf{p}?x; Q], \Sigma, \mathscr{P}\rangle \xrightarrow{\mathsf{p}.v \texttt{->} \mathsf{q}} \langle \mathsf{p}[P] \mid \mathsf{q}[Q], \Sigma\left[\mathsf{q}.x \mapsto v\right], \mathscr{P}\rangle} \ [\textsc{Com}]$$

$$\frac{j \in I}{\langle \mathsf{p}[\mathsf{q} \oplus \textsc{l}_j; P] \mid \mathsf{q}[\mathsf{p} \,\&\{\textsc{l}_i : P_i\}_{i \in I}; Q], \Sigma, \mathscr{P}\rangle \xrightarrow{\mathsf{p} \texttt{->} \mathsf{q}.\textsc{l}_j} \langle \mathsf{p}[P] \mid \mathsf{q}[P_j \fatsemi Q], \Sigma, \mathscr{P}\rangle} \ [\textsc{Sel}]$$

$$\frac{\Sigma(\mathsf{p}) \vdash e \downarrow v}{\langle \mathsf{p}[x := e; P], \Sigma, \mathscr{P}\rangle \xrightarrow{\tau @ \mathsf{p}} \langle \mathsf{p}[P], \Sigma\left[\mathsf{p}.x \mapsto v\right], \mathscr{P}\rangle}\ [\textsc{Local}]$$

$$\frac{i = 1 \text{ if } \Sigma(\mathsf{p}) \vdash e \downarrow \mathtt{true},\ i = 2 \text{ otherwise}}{\langle \mathsf{p}[\mathsf{if}\ e\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2;\ Q], \Sigma, \mathscr{P}\rangle \xrightarrow{\tau @ \mathsf{p}} \langle \mathsf{p}[P_i \fatsemi Q], \Sigma, \mathscr{P}\rangle}\ [\textsc{Cond}]$$

$$\frac{X(\vec{\mathsf{r}}) = Q \ \in\ \mathscr{P}}{\langle \mathsf{p}[X(\vec{\mathsf{q}}); P], \Sigma, \mathscr{P}\rangle \xrightarrow{\tau @ \mathsf{p}} \langle \mathsf{p}[Q[\vec{\mathsf{q}}/\vec{\mathsf{r}}] \fatsemi P], \Sigma, \mathscr{P}\rangle}\ [\textsc{Call}]$$

$$\frac{\langle N, \Sigma, \mathscr{P}\rangle \xrightarrow{\mu} \langle N', \Sigma', \mathscr{P}\rangle}{\langle N \mid M, \Sigma, \mathscr{P}\rangle \xrightarrow{\mu} \langle N' \mid M, \Sigma', \mathscr{P}\rangle}\ [\textsc{Par}]$$