Skriftlig Eksamen

DM861 Concurrency Theory

Institut for Matematik og Datalogi, Syddansk Universitet, Odense

2019-01-08

Instructions

Modality The following conventions are used/suggested if you wish to hand in answers in pure text 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''`).
- Single and double arrows are `--->` and `===>` (or variations thereof); the mapping arrow $\mapsto$ as `|-->`.
- The right triangle $\triangleright$ is `|>`.
- The downarrow evaluation ($e \downarrow v$) is `\eval`.
- Structural precongruence and equivalence are `<=` and `===`.
- Often common symbols are rendered by the name of their (standard) latex macros, e.g. σ is `\sigma`.
- $\oplus$ 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).

Supplementary material You may only consult the material in the Appendix.

$$\frac{}{\mathsf{move}(x_1, x_2, \mathsf{a})} \qquad \frac{}{\mathsf{move}(x_1, x_3, \mathsf{a})} \qquad \frac{}{\mathsf{move}(x_2, x_2, \mathsf{c})} \qquad \frac{}{\mathsf{move}(x_2, x_3, \mathsf{b})} \qquad \frac{}{\mathsf{move}(x_3, x_4, \mathsf{a})}$$

$$\frac{\mathsf{move}(X, Y, w)}{\mathsf{word}(X, Y, w)}\ [\textsc{Dir}] \qquad \frac{\mathsf{word}(X, Y, w) \quad \mathsf{word}(Y, Z, w')}{\mathsf{word}(X, Z, ww')}\ [\textsc{Trans}]$$

$$\frac{}{\mathsf{final}(x_3)} \qquad \frac{\mathsf{word}(X, Y, w) \quad \mathsf{final}(Y)}{\mathsf{accept}(X, w)}\ [\textsc{Lang}]$$

System 1: Word automaton.

Question 1

Consider the rules in System 1 where

- $\{x_1, x_2, x_3, x_4\}$ is a fixed set of symbols hereafter referred to as *states*;
- $\{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$ is a fixed set of characters symbols (disjoint from that of states);
- X, Y, Z are schematic variables ranging over states;
- c is a schematic variable ranging over characters;
- w, w' are schematic variables ranging over finite strings of characters (words);
- ε denotes the empty string;
- the expression ww' denotes the concatenation of strings w and w'.

Answer the following questions and motivate your answers formally.

(a) Are $\mathsf{accept}(x_3, \mathsf{a})$, $\mathsf{accept}(x_1, \mathsf{ab})$, $\mathsf{accept}(x_1, \mathsf{abc})$, and $\mathsf{accept}(x_1, \mathsf{acb})$ derivable using the rules in System 1?

(b) Is rule [Empty] below admissible in System 1?

$$\frac{\mathsf{final}(X)}{\mathsf{accept}(X, \varepsilon)}\ [\textsc{Empty}]$$

(c) Is rule [Step] below derivable in System 1?

$$\frac{\mathsf{move}(X, Y, c) \quad \mathsf{word}(Y, Z, w)}{\mathsf{word}(X, Z, cw)}\ [\textsc{Step}]$$

(d) Is rule [Head1] below admissible in System 1?

$$\frac{\mathsf{word}(X, Z, cw)}{\mathsf{move}(X, Y, c)}\ [\textsc{Head1}]$$

(e) Is rule [Head2] below admissible in System 1?

$$\frac{\mathsf{word}(X, Y, c) \quad \mathsf{word}(Y, Z, w)}{\mathsf{move}(X, Y, c)}\ [\textsc{Head2}]$$

Question 2

(a) Write a projectable choreography in the language of Recursive Choreographies (see Appendix) for:

- p sends a product name to q;
- p sends a product quantity to q;
- q asks r whether the product name is in stock;
- r replies to q with the quantity of product that is available;
- q decides:
  - if there 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 voucher is not valid (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:
    - q sends a voucher to p;
    - p sends the shipping address to r.
  - otherwise
    - nothing happens

Question 3

Show all reduction chains for the configuration $\langle C, \sigma \rangle$ in the language of Recursive Choreographies (you can omit $\mathscr{C}$), where C is

```
(if p.f then
  p -> q [ok];
  q.f1 -> p.g1;
  p.f2 -> q.g2;
  0
else
  p -> q [ko];
  p.f1 -> q.g1;
  q.f2 -> p.g2;
  0);0
```

and σ is such that $f(\sigma(\mathsf{p})) = \mathtt{true}$.

Question 4

Let C be the choreography `r -> s; r -> p; s -> p; 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.

Question 5

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

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

```
p ▷ s?;r!;0  | q ▷ s!;0  | r ▷ s!;p?;0  | s ▷ r?;p!;q?;0
```

(a) Are they operationally correspondent? Motivate your answer informally.

(b) Give a formal argument for your answer to the previous question. (Hint: use reduction chains.)

Question 6

Under which assumption does procedure Book, defined below in the language of Recursive Choreographies, terminate? Notice that `q.changePrices` may change both the values computed by `q.price(x)` and `q.discount(x)` in the next iteration.

```
Book(p,q,b) =
  p.date -> q.x;
  if q.free(x) then
    q -> p[ok];
    p -> b[ok];
    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;
      0
    else
      p -> b[ko];
      p -> q[ko];
      q.changePrices;
      Book(p,q,b);
      0
  else
    q -> p[ko];
    p -> b[ko];
    p.changeDate;
    Book(p,q,b);
    0;
  0
```

Question 7

(a) Is this network deadlock-free (in the network language for Simple Concurrent Choreographies)? Answer yes or no.

```
p ▷ s?;0  | q ▷ t!;0 |  r ▷ s!;0  | s ▷ r?;p!;0 | t ▷ q?;s!;0
```

(b)
- If it is deadlock-free, provide a choreography whose EPP is the network above.
- If it is not deadlock-free, show a reduction chain of the network above that leads to a deadlock.

Question 8

(a) Is this network deadlock-free (in the network language for Recursive Choreographies)? Answer yes or no.

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

(b)
- If it is deadlock-free, provide a choreography whose EPP is the network above.
- If it is not deadlock-free, show a reduction chain of the network above that leads to a deadlock.

Since there are no procedures, you can omit the sets of procedures for both choreographies and networks.

Question 9

For each of the following statements, write whether the statement holds or not. Informally motivate your answers by referring to the definition of $\preceq$ for Simple Concurrent Choreographies.

(a) `r -> s; p -> s; 0` $\preceq$ `p -> s; r -> s; 0`

(b) `p -> q; r -> s; 0` $\preceq$ `r -> s; p -> q; 0`

(c) `t -> p; s -> q; r -> s; 0` $\preceq$ `r -> s; s -> q; t -> p; 0`

(d) `r -> s; p -> s; t -> q; 0` $\preceq$ `t -> q; r -> s; p -> s; 0`

Simple Concurrent Choreographies

The syntax of the simple choreography language is defined by the grammar below.

$$C ::= \mathsf{p} \twoheadrightarrow \mathsf{q}; C \mid 0$$

The reduction semantics is defined by the derivation rules below.

$$\frac{}{\mathsf{p} \twoheadrightarrow \mathsf{q}; C \to C}\ [\textsc{Com}] \qquad \frac{C \preceq C_1 \quad C_1 \to C_2 \quad C_2 \preceq C'}{C \to C'}\ [\textsc{Struct}]$$

$$\frac{\{\mathsf{p},\mathsf{q}\} \# \{\mathsf{r},\mathsf{s}\}}{\mathsf{p} \twoheadrightarrow \mathsf{q}; \mathsf{r} \twoheadrightarrow \mathsf{s} \quad \equiv \quad \mathsf{r} \twoheadrightarrow \mathsf{s}; \mathsf{p} \twoheadrightarrow \mathsf{q}}\ [\textsc{Swap}]$$

The collection of network terms and simple processes is given by the following grammar:

$$N ::= \mathsf{p} \triangleright B \mid N \mid N \mid 0$$
$$B ::= \mathsf{p}!; B \mid \mathsf{p}?; B \mid 0$$

Their reduction semantics is defined by the derivation rules below.

$$\frac{}{\mathsf{p} \triangleright \mathsf{q}!; B \ \mid\ \mathsf{q} \triangleright \mathsf{p}?; B' \quad \to \quad \mathsf{p} \triangleright B \ \mid\ \mathsf{q} \triangleright B'}\ [\textsc{Com}]$$

$$\frac{N_1 \to N_1'}{N_1 \mid N_2 \ \to\ N_1' \mid N_2}\ [\textsc{Par}] \qquad \frac{N \preceq N_1 \quad N_1 \to N_2 \quad N_2 \preceq N'}{N \to N'}\ [\textsc{Struct}]$$

$$\frac{}{(N_1 \mid N_2) \mid N_3 \ \equiv\ N_1 \mid (N_2 \mid N_3)}\ [\text{PA}]$$

$$\frac{}{N_1 \mid N_2 \ \equiv\ N_2 \mid N_1}\ [\text{PC}] \qquad \frac{}{N \mid 0 \ \preceq\ N}\ [\text{GCN}] \qquad \frac{}{\mathsf{p} \triangleright 0 \ \preceq\ 0}\ [\text{GCP}]$$

Recursive Choreographies

The syntax of the recursive choreography language is defined by the grammar below.

$$C ::= I; C \mid 0$$
$$I ::= \mathsf{p}.f \twoheadrightarrow \mathsf{q}.g \mid \mathsf{p} \twoheadrightarrow \mathsf{q}[l] \mid \mathsf{p}.f \mid \mathsf{if}\ \mathsf{p}.f\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2 \mid X(\tilde{\mathsf{p}}) \mid 0$$
$$\mathscr{C} ::= X(\tilde{\mathsf{p}}) = C, \mathscr{C} \mid \emptyset$$

The reduction semantics is defined by the derivation rules below.

$$\frac{f(\sigma(\mathsf{p})) \downarrow v \quad g(\sigma(\mathsf{q}), v) \downarrow u}{\langle \mathsf{p}.f \twoheadrightarrow \mathsf{q}.g; C, \sigma\rangle \to_{\mathscr{C}} \langle C, \sigma[\mathsf{q} \mapsto u]\rangle}\ [\textsc{Com}] \qquad \frac{}{\langle \mathsf{p} \twoheadrightarrow \mathsf{q}[l]; C, \sigma\rangle \to_{\mathscr{C}} \langle C, \sigma\rangle}\ [\textsc{Sel}]$$

$$\frac{f(\sigma(\mathsf{p})) \downarrow v}{\langle \mathsf{p}.f; C, \sigma\rangle \to_{\mathscr{C}} \langle C, \sigma[\mathsf{p} \mapsto v]\rangle}\ [\textsc{Local}] \qquad \frac{i = 1 \text{ if } f(\sigma(\mathsf{p})) \downarrow \texttt{true},\ i = 2 \text{ otherwise}}{\langle \mathsf{if}\ \mathsf{p}.f\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2; C, \sigma\rangle \to_{\mathscr{C}} \langle C_i; C, \sigma\rangle}\ [\textsc{Cond}]$$

$$\frac{C \preceq_{\mathscr{C}} C_1 \quad \langle C_1, \sigma\rangle \to_{\mathscr{C}} \langle C_2, \sigma'\rangle \quad C_2 \preceq_{\mathscr{C}} C'}{\langle C, \sigma\rangle \to_{\mathscr{C}} \langle C', \sigma'\rangle}\ [\textsc{Struct}] \qquad \frac{\mathsf{procs}(I) \,\#\, \mathsf{procs}(I')}{I; I' \quad \equiv_{\mathscr{C}} \quad I'; I}\ [\text{I-I}]$$

$$\frac{\mathsf{p} \notin \mathsf{procs}(I)}{I; \mathsf{if}\ \mathsf{p}.f\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2 \quad \equiv_{\mathscr{C}} \quad \mathsf{if}\ \mathsf{p}.f\ \mathsf{then}\ (I; C_1)\ \mathsf{else}\ (I; C_2)}\ [\text{I-}\textsc{Cond}]$$

$$\frac{\mathsf{p} \notin \mathsf{procs}(I) \quad I \neq 0}{\mathsf{if}\ \mathsf{p}.f\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2; I \quad \equiv_{\mathscr{C}} \quad \mathsf{if}\ \mathsf{p}.f\ \mathsf{then}\ (C_1; I)\ \mathsf{else}\ (C_2; I)}\ [\text{}\textsc{Cond}\text{-I}]$$

$$\frac{}{0; C \preceq_{\mathscr{C}} C}\ [\textsc{GCNil}] \qquad \frac{X(\tilde{\mathsf{q}}) = C \ \in \mathscr{C}}{X(\tilde{\mathsf{p}}) \preceq_{\mathscr{C}} C[\tilde{\mathsf{p}}/\tilde{\mathsf{q}}]}\ [\textsc{Unfold}]$$

The collection of network terms and recursive processes is given by the following grammar:

$$\begin{aligned} N &::= \mathsf{p} \triangleright_v B \mid N \mid N \mid 0 \\ B &::= \mathsf{p}!f; B \mid \mathsf{p}?f; B \mid \mathsf{p} \oplus l; B \mid \mathsf{p} \,\&\{l_i : B_i\}_{i \in I}; B \mid \mathsf{if}\ f\ \mathsf{then}\ B_1\ \mathsf{else}\ B_2; B \mid 0; B \mid X(\tilde{\mathsf{p}}) \mid 0 \\ \mathscr{B} &::= X(\tilde{\mathsf{p}}) = B, \mathscr{B} \mid \emptyset \end{aligned}$$

The reduction semantics of networks of recursive processes is defined by the derivation rules below.

$$\frac{f(v) \downarrow v' \quad g(u, v') \downarrow u'}{\mathsf{p} \triangleright_v \mathsf{q}!f; B \ \mid\ \mathsf{q} \triangleright_u \mathsf{p}?g; B' \quad \to_{\mathscr{B}} \quad \mathsf{p} \triangleright_v B \ \mid\ \mathsf{q} \triangleright_{u'} B'}\ [\textsc{Com}]$$

$$\frac{j \in I}{\mathsf{p} \triangleright_v \mathsf{q} \oplus l_j; B \ \mid\ \mathsf{q} \triangleright_u \mathsf{p} \,\&\{l_i : B_i\}_{i \in I}; B' \quad \to_{\mathscr{B}} \quad \mathsf{p} \triangleright_v B \ \mid\ \mathsf{q} \triangleright_u B_j; B'}\ [\textsc{Sel}]$$

$$\frac{i = 1 \text{ if } f(v) \downarrow \mathtt{true},\ i = 2 \text{ otherwise}}{\mathsf{p} \triangleright_v (\mathsf{if}\ f\ \mathsf{then}\ B_1\ \mathsf{else}\ B_2)\,; B \to_{\mathscr{B}} \mathsf{p} \triangleright_v B_i; B}\ [\textsc{Cond}]$$

$$\frac{N_1 \to_{\mathscr{B}} N_1'}{N_1 \mid N_2 \ \to_{\mathscr{B}} \ N_1' \mid N_2}\ [\textsc{Par}] \qquad \frac{N \preceq_{\mathscr{B}} N_1 \quad N_1 \to_{\mathscr{B}} N_2 \quad N_2 \preceq_{\mathscr{B}} N'}{N \to_{\mathscr{B}} N'}\ [\textsc{Struct}]$$

$$\frac{}{(N_1 \mid N_2) \mid N_3 \ \equiv_{\mathscr{B}} \ N_1 \mid (N_2 \mid N_3)}\ [\text{PA}] \qquad \frac{}{0; B \preceq_{\mathscr{B}} B}\ [\text{GCB}]$$

$$\frac{}{N_1 \mid N_2 \ \equiv_{\mathscr{B}} \ N_2 \mid N_1}\ [\text{PC}] \qquad \frac{}{N \mid 0 \ \preceq_{\mathscr{B}} \ N}\ [\text{GCN}] \qquad \frac{}{\mathsf{p} \triangleright_v 0 \ \preceq_{\mathscr{B}} \ 0}\ [\text{GCP}]$$

$$\frac{X(\tilde{\mathsf{q}}) = B \ \in \mathscr{B}}{X(\tilde{\mathsf{p}}) \preceq_{\mathscr{B}} B[\tilde{\mathsf{p}}/\tilde{\mathsf{q}}]}\ [\textsc{Unfold}]$$