MSc in Mathematics and Foundations of Computer Science

DISTRIBUTED PROCESSES, TYPES AND PROGRAMMING

Michaelmas Term 2024

Submission deadline: 12 noon, Wednesday 8th January 2025, via Inspera.

There is a total of 100 marks available for this paper, you should attempt all parts of the paper.

NB: You must not discuss this examination paper with anyone.

Please submit answers to all questions as a single .pdf file and typeset with a font size at least 11 point and use A4 paper format with all margins at least 2 cm.

TURN OVER

Distributed Processes, Types and Programming

There is a total of 100 marks available for this paper, you should attempt all parts of the paper.

Please submit answers to all questions as a single .pdf file and typeset with a font size at least 11 point and use A4 paper format with all margins at least 2 cm.

Question 1

(a) Give the free names and free variables of the following processes.

(1) $(\nu\, d)(x(y).\overline{z}\langle d\rangle) \mid !e(y).\overline{y}\langle y\rangle \mid !d(y).(\nu\, c)\overline{y}\langle c\rangle$

(2) $c(z).(\nu\, e)(\nu\, a)\overline{e}\langle z\rangle \mid ((\nu\, a, b)y(x).\overline{b}\langle x\rangle \mid \overline{b}\langle a\rangle \mid \overline{e}\langle x\rangle)$

(4 marks)

(b) This question is about the monadic asynchronous π-calculus. Assume $\mathcal{P}$ denotes the set of processes and $\mathbb{N}$ denotes the set of natural numbers. The function $\mathsf{size} : \mathcal{P} \to \mathbb{N}$ which returns a size of a given process is defined by:

$\mathsf{size}(\mathbf{0}) = \mathsf{size}(\overline{u}\langle v\rangle) = 1$, $\mathsf{size}(u(x).P) = 1 + \mathsf{size}(P)$,
$\mathsf{size}((\nu\, a)P) = \mathsf{size}(!P) = 1 + \mathsf{size}(P)$ and $\mathsf{size}(P \mid Q) = \mathsf{size}(P) + \mathsf{size}(Q)$.

(1) Define the size of the context C (denoted by $\mathsf{size}(C)$) $\mathsf{size} : \mathcal{P} \cup \{-\} \to \mathbb{N}$ assuming $\mathsf{size}(-) = 1$:

$$C ::= \quad - \quad \mid \quad C \mid P \quad \mid \quad P \mid C \quad \mid \quad (\nu\, a)C$$

You can use $\mathsf{size}(P)$ defined above.

(2) Assume

$$P \;=\; a(x).(a(x).\overline{a}\langle x\rangle \mid \overline{a}\langle x\rangle) \qquad \text{and} \qquad P' \;=\; a(x).(\overline{a}\langle x\rangle \mid \overline{a}\langle x\rangle)$$

Are P and P' reduction congruent? More specifically:

- If P and P' are reduction congruent, prove $P \cong P'$;
- Otherwise prove $P \ncong P'$, giving the minimum size of context C which differentiates P from P'.

You can use the equivalence laws given in the slides.

(3) Assume

$$Q \;= !a(x).(a(x).\overline{a}\langle x\rangle \mid \overline{a}\langle x\rangle) \qquad Q' \;= !a(x).(\overline{a}\langle x\rangle \mid \overline{a}\langle x\rangle)$$

Are Q and Q' reduction congruent? More specifically:

- If Q and Q' are reduction congruent, prove $Q \cong Q'$;
- Otherwise prove $Q \ncong Q'$, giving the minimum size of context C which differentiates Q from Q'.

You can use the equivalence laws given in the slides.

(16 marks)

(c) This is a question about the asynchronous monadic π-calculus. Assume two statements:

(i) if $P \downarrow a$ for some a, then $Q \downarrow a$; and if $Q \downarrow a$ for some a, then $P \downarrow a$

(ii) if $P \Downarrow a$ for some a, then $Q \Downarrow a$; and if $Q \Downarrow a$ for some a, then $P \Downarrow a$

Does (i) imply (ii)? I.e., if a pair of P and Q satisfies (i), then a pair of P and Q satisfies (ii)? If it is true, prove that (i) implies (ii). If not, find out a counterexample.

Conversely, does (ii) imply (i)? If it is true, prove that (ii) implies (i). If not, find out a counterexample. (5 marks)

Question 2

Suppose the following direct encoding $[\![\cdot]\!]$ from the polyadic synchronous π-calculus to the monadic asynchronous π-calculus in Class Sheet 2:

$$[\![u(x_1,..,x_n).P]\!] = u(y).(\nu\, d)(\overline{y}\langle d\rangle \mid d(x_1).(\overline{y}\langle d\rangle \mid d(x_2).(...(\overline{y}\langle d\rangle \mid d(x_n).[\![P]\!])...)))$$

where $y \notin \mathit{fv}(P)$, and $d \notin \mathit{fn}(P)$.

$$[\![\overline{u}\langle v_1,..,v_n\rangle.P]\!] = (\nu\, c)(\overline{u}\langle c\rangle \mid c(y_1).(\overline{y_1}\langle v_1\rangle \mid c(y_2).(\overline{y_2}\langle v_2\rangle \mid ... \mid c(y_n).(\overline{y_n}\langle v_n\rangle \mid [\![P]\!])...)))$$

where $y_i \notin \mathit{fv}(P)$ $(1 \leq i \leq n)$ and $c \notin \mathit{fn}(P)$.

We assume the polyadic synchronous π-calculus is *typed*, i.e., we assume the identifier u always sends and receives a vector of the same arity. For example, Q is not typed but R is typed assuming R_1, R_2 and R_3 are also typed.

$$\begin{array}{rcl} Q & = & a(x).Q_1 \mid a(x_1,x_2).Q_2 \mid \overline{a}\langle b\rangle.Q_3 \\ R & = & a(x_1,x_2).R_1 \mid a(x_1,x_2).R_2 \mid \overline{a}\langle b_1,b_2\rangle.R_2 \end{array}$$

Recall the formal definition of encodability criteria from Page 5 in the slides "Correctness of Encodings and Separation Results".

(a) Explain the reason (with an example) why the polyadic π-calculus must be typed to prove the above encoding satisfies the criteria. (5 marks)

(b) Assuming the polyadic synchronous π-calculus is typed, prove the above encoding satisfies the encodability criteria. (20 marks)

TURN OVER

Question 3

(a) This question is about *binary (2-party)* session types.

Assume the following session types, S_1 and S_2. Do they satisfy $S_1 \leqslant S_2$? If so, prove $S_1 \leqslant S_2$ using the method of Solution V2 or Solution V3 from the slides. Otherwise show $S_1 \not\leqslant S_2$ using the method of Solution V2 or Solution V3 from the slides.

$$S_1 = \mu \mathbf{t}.\mathbf{Alice}?[\mathtt{int}];\mathbf{Alice}\& \left\{ \begin{array}{l} finish : \mathtt{end} \\ check : \mathbf{Alice}![\mathtt{nat}];\mathtt{end} \\ loop : \mathbf{t} \\ quit : \mathbf{Alice}![\mathtt{bool}];\mathtt{end} \end{array} \right\}$$

$$S_2 = \mathbf{Alice}?[\mathtt{int}];\mu \mathbf{t}.\mathbf{Alice}\& \left\{ \begin{array}{l} loop : \mathbf{Alice}?[\mathtt{int}];\mathbf{Alice}\& \left\{ \begin{array}{l} loop : \mathbf{Alice}?[\mathtt{int}];\mathbf{t} \\ check : \mathbf{Alice}![\mathtt{nat}];\mathtt{end} \end{array} \right\} \\ finish : \mathtt{end} \end{array} \right\}$$

In your proof, you can omit **Alice** from the syntax of types. (15 marks)

(b) (1) The following global type, G_{bad}, is not well-formed under either plain or full merging.

$$G_{\text{bad}} = \mu \mathbf{t}.\mathbf{A} \to \mathbf{B} \left\{ \begin{array}{l} accept : \mathbf{C} \to \mathbf{B} \left\{ \begin{array}{l} ping : \mathbf{B} \to \mathbf{A}\,\{pong : \mathbf{t}\} \\ stop : \mathtt{end} \end{array} \right\} \\ reject : \mathbf{C} \to \mathbf{B}\,\{ping : \mathbf{B} \to \mathbf{A}\,\{stop : \mathtt{end}\}\} \end{array} \right\}$$

Explain why it is not well-formed.

(2) Amend G_{bad} to make it well-formed under full merging but *not* under plain merging. There might be several ways to fix G_{bad}. You only need to show one fix.

Then give the projection of the global type you fixed onto **A**, **B** and **C**.

(10 marks)

Question 4

(a) Explain with an example why the side condition "$\mu \mathbf{t}.G$ is closed" is required for the projection of the recursive type rule given below.

$$\begin{array}{rcl} \mu \mathbf{t}.G \upharpoonright \mathbf{r} & = & \begin{cases} \mathtt{end} & \mathbf{r} \notin \mathsf{pt}(G) \text{ and } \mu \mathbf{t}.G \text{ is closed} \\ \mu \mathbf{t}.(G \upharpoonright \mathbf{r}) & \text{otherwise} \end{cases} \\ \mathbf{t} \upharpoonright \mathbf{r} & = & \mathbf{t} \\ \mathtt{end} \upharpoonright \mathbf{r} & = & \mathtt{end} \end{array}$$

(5 marks)

(b) Consider the following two global types:

$$\begin{array}{rcl} G_1 = & \mu \mathbf{t}.\mathbf{A} \to \mathbf{B} \left\{ \begin{array}{l} \mathit{accept} : \mathbf{C} \to \mathbf{D} : [\mathtt{nat}]; \mathbf{C} \to \mathbf{D} : [\mathtt{nat}]; \mathbf{t} \\ \mathit{reject} : \mu \mathbf{t}'.\mathbf{C} \to \mathbf{D} : [\mathtt{nat}]; \mathbf{t}' \end{array} \right\} \\ G_2 = & \mu \mathbf{t}.\mathbf{A} \to \mathbf{B} \left\{ \begin{array}{l} \mathit{accept} : \mathbf{C} \to \mathbf{D} : [\mathtt{nat}]; \mathbf{t} \\ \mathit{reject} : \mu \mathbf{t}'.\mathbf{C} \to \mathbf{D} : [\mathtt{nat}]; \mathbf{C} \to \mathbf{D} : [\mathtt{nat}]; \mathbf{t}' \end{array} \right\} \end{array}$$

(1) Prove that they represent the same protocol using the transition relation $G \stackrel{\alpha}{\Rightarrow} G'$ defined in *Global Type Semantics* in "Subject Reduction Proof" available from the course web page.

(2) G_1 and G_2 are not projectable under either plain or full merging given in the slides. Define the projection rules which enable to project G_1 and G_2, and justify your rules using G_1 and G_2.

(20 marks)

LAST PAGE