Concurrency Theory

Prof. Dr. Peter Thiemann — University of Freiburg
Marius Weidner, Leonardo Mieschendahl — Winter 2025

Sheet 1

Due: Monday, 2025-11-03

Important Information:

- Exercises are ungraded and do *not* need to be submitted.
- If you have questions, please post a message in the dedicated chat.
- The solutions will be discussed in the tutorial sessions.

Aufgabe 1.1 (Atomic Parallel Execution)

What are the possible values of x at the end of the execution of the following program:

$$\texttt{x:=10; ((x:=x*2; x:=x-11; x:=x+2)} \parallel \texttt{x:=x-5)}$$

You can assume atomic operations.

Aufgabe 1.2 (Labeled Transition Systems)

Consider the following LTS:

$$\begin{tikzcd} 1 \arrow[r, "a"] & 2 \arrow[d, "a"] \\ 4 \arrow[u, "a"] & 3 \arrow[l, "a"] \end{tikzcd}$$

- Formally define the LTS.
- What is the reflexive closure of the binary relation $\xrightarrow{a}$? (just draw it)
- What is the symmetric closure of the binary relation $\xrightarrow{a}$? (just draw it)
- What is the transitive closure of the binary relation $\xrightarrow{a}$? (just draw it)

Aufgabe 1.3 (Informal Specification to CCS)

Consider the following CCS definition of a `coffee` machine:

$$\texttt{CM} \doteq \texttt{coin}.\overline{\texttt{coffee}}.\texttt{CM}$$

Give a CCS definition that describes a `coffee` machine which can take money without returning `coffee`, and which can fail at any time.

Aufgabe 1.4 (Relating CCS and LTS)

Consider an LTS with a finite number of states and action labels.

- Does this imply that the $\xrightarrow{a}$ set is finite?

- Draw an example of an LTS with 4 states and two transitions.
- How can your example be described by a sequential fragment of CCS (i.e. no parallel execution).
- Show that in general, any finite LTS can be described by using a sequential fragment of CCS.

Aufgabe 1.5 (Formal CCS Semantics)
By using the SOS rules for CCS prove the existence of the following transitions (assume that $A \doteq b.a.B$):

- $(A \parallel \bar{b}.Nil) \setminus \{b\} \xrightarrow{\tau} (a.B \parallel Nil) \setminus \{b\}$
- $(A \parallel \bar{b}.a.B) + (b.A)[a/b] \xrightarrow{\bar{b}} (A \parallel a.B)$
- $(A \parallel \bar{b}.a.B) + (\bar{b}.A)[a/b] \xrightarrow{\bar{a}} A[a/b]$

Aufgabe 1.6 (CSS to LTS using Formal Semantics)
Consider the following CCS defining equations:

$$\texttt{CM} \doteq \texttt{coin}.\overline{\texttt{coffee}}.\texttt{CM}$$

$$\texttt{CS} \doteq \overline{\texttt{pub}}.\overline{\texttt{coin}}.\texttt{coffee}.\texttt{CS}$$

$$\texttt{Uni} \doteq (\texttt{CM} \parallel \texttt{CS}) \setminus \{\texttt{coin}, \texttt{coffee}\}$$

Use the rules of the SOS semantics for CCS to derive the labelled transition system for the process `Uni` defined above. The proofs can be omitted and a drawing of the resulting LTS is enough.

Aufgabe 1.7 (Infinite LTS)
Draw (part of) the labelled transition system for the process constant A defined by

$$A \doteq (a.A) \setminus \{b\}.$$

The resulting LTS should have infinitely many reachable states. Can you think of a CCS term that generates a finite LTS and intuitively has the same *behaviour* as A?

Aufgabe 1.8 (LTS Isomorphisms and Trace Equivalence)

(a) Draw the transition graph for the process name $\texttt{Mutex}_1$, whose behaviour is given by the following defining equations.

$$\begin{aligned} \texttt{Mutex}_1 &\doteq (\texttt{User} \parallel \texttt{Sem}) \setminus \{p, v\} \\ \texttt{User} &\doteq \bar{p}.\texttt{enter}.\texttt{exit}.\bar{v}.\texttt{User} \\ \texttt{Sem} &\doteq p.v.\texttt{Sem} \end{aligned}$$

(b) Draw the transition graph for the process name $\texttt{Mutex}_2$ whose behaviour is given by the defining equation

$$\texttt{Mutex}_2 \doteq ((\texttt{User} \parallel \texttt{Sem}) \parallel \texttt{User}) \setminus \{p, v\}$$

where `User` and `Sem` are defined as before. Would the behaviour of the process change if `User` was defined as

$$\texttt{User} \doteq \bar{p}.\texttt{enter}.\bar{v}.\texttt{exit}.\texttt{User}$$

(c) Draw the transition graph for the process name `FMutex` whose behaviour is given by the defining equation

$$\texttt{FMutex} \doteq ((\texttt{User} \parallel \texttt{Sem}) \parallel \texttt{FUser}) \setminus \{p, v\}$$

where `User` and `Sem` are defined as before, and the behaviour of `FUser` is given by the defining equation

$$\texttt{FUser} \doteq \bar{p}.\texttt{enter}.(\texttt{exit}.\bar{v}.\texttt{FUser} + \texttt{exit}.\bar{v}.Nil)$$

Do you think that $\texttt{Mutex}_2$ and `FMutex` are offering the same behaviour? Can you argue informally for your answer?

(d) Are the LTS of $\texttt{Mutex}_2$ and `FMutex` ismorphic?

(e) Are $\texttt{Mutex}_2$ and `FMutex` trace equivalent?