Concurrency Semantics
Exercises 1

Prof. Nestmann, 2005

1. Simulation

Let $(\mathcal{Q}, \mathcal{T})$ be the LTS over $\mathcal{A} = \{ b, c \}$ where

$$\begin{array}{rcl} \mathcal{Q} & = & \{ 1, 2, 3, 4, 5, 6, 7, 8, 9 \} \\ \mathcal{T} & = & \{ (1, b, 2), (1, c, 3), (4, b, 5), (6, b, 7), (6, c, 8), (6, c, 9) \} \end{array}$$

- We call pairs in $\mathcal{Q} \times \mathcal{Q}$ *trivial*
  1. if they are an element of the identity relation on $\mathcal{Q}$, or
  2. if they are an element of $\{2, 3, 5, 7, 8, 9\} \times \{2, 3, 5, 7, 8, 9\}$.
- We call simulations *trivial* if they
  1. are empty
  2. contain *only* trivial pairs
  3. contain at least one trivial pair that is not reachable from a contained non-trivial one

Find *all non-trivial* simulations in $(\mathcal{Q}, \mathcal{T})$. How many are there?
(Hint: there are more than you might expect ...)

2. Bisimulation

Let $(\mathcal{Q}, \mathcal{T})$ be an arbitrary LTS.
Prove that $\sim$ is an equivalence relation.

3. Operational Semantics

Milner's Scheduler Example and Exercise (§3.6, Exercise 3.15).

A set of n processes $P_i, 0 \leq i \leq n{-}1$ is to be scheduled as follows:

- P_i starts a task by sync'ing on a_i with the scheduler.
- P_i completes a task by sync'ing on b_i with the scheduler.

Concurrency is allowed:

- Tasks of different P_i *may* run at the same time.

There is a mutual exclusion property to be respected:

- Each P_i *must not* run two tasks at a time.
- For each i, a_i and b_i must occur cyclically.

The scheduling of start permissions shall be *round-robin*:

- The a_i are required to occur cyclically (initially, 0 starts)

The overall systen shall provide *maximal "progress"*:

- the scheduling must permit any of the "buttons" to be pressed at any time provided the other properties are not violated.

The specification can be formalized as sequential non-deterministic process.
Let $i \in \{0 \ldots, n{-}1\}$. Let $X \subseteq \{0 \ldots, n{-}1\}$. Then $\mathsf{S}_{\mathsf{i,X}}(\vec{a}, \vec{b})$, defined by

$$\mathsf{S}_{i,X} \stackrel{\text{def}}{=} \begin{cases} \sum_{j \in X} b_j.\mathsf{S}_{i,X-j} & (i \in X) \\ \sum_{j \in X} b_j.\mathsf{S}_{i,X-j} + a_i.\mathsf{S}_{(i+1) \bmod n, X \cup i} & (i \notin X) \end{cases}$$

represents a scheduler, where i is next to have the start permission, and where every $j \in X$ is currently running. Initially:

$$\mathsf{Scheduler}_n \stackrel{\text{def}}{=} \mathsf{S}_{0,\emptyset}$$

Tasks:

1. Draw the transition graph for $n = 2$.
2. Argue why the scheduler is never deadlocked.
3. Understand the difference in behavior when dropping the case for $i \in X$.