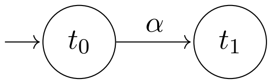
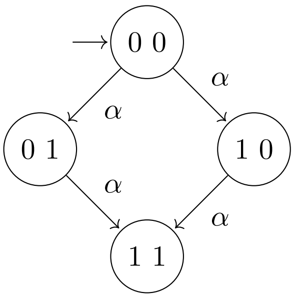
Aaron Grabowy: 345766
Timo Bergerbusch: 344408
Felix Linhart: 318801

Exercise 1

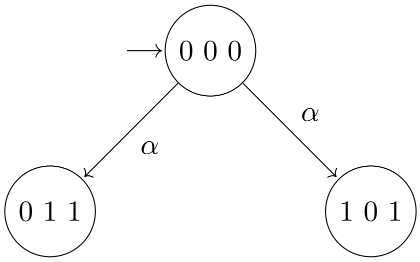
a)

Let TS_1, TS_2 and TS_3 be: 

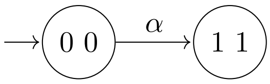
Further let $H = \emptyset$ and $H' = \{\alpha\}$. A node contains the shorthand notation of for example “0 0” to denote that TS_1 and TS_2 (in the second case TS_2 and TS_3) are in state t_0. Analogously for “0 0 0” and TS_1, TS_2 and TS_3. Then we can construct $TS_4 \coloneqq (TS_1||_H TS_2)||_{H'} TS_3$ as:



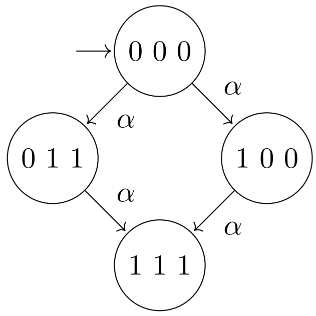
$||_{H'} TS_3$ $\Rightarrow$



Now we construct $TS_4' \coloneqq TS_1||_H (TS_2||_{H'} TS_3)$:

$TS_1||_H$ 

$\Rightarrow$



As we can see $TS_4 \neq TS_4'$ and therefore in general the handshaking $||_H$ is not associative.

b)

In order two proof that the bijection $f_{\approx}$ preserves the stated transition relation, we consider 3 base cases:

i) $\alpha \in Act_1 \setminus (Act_2 \cup Act_3)$

If α is neither contained in Act_2 nor Act_3 then for every state $\langle\langle s_1, s_2\rangle s_3\rangle$ and every transition $(s_1, \alpha, s_1') \in\to_1$ there exists a transition

$$(\langle\langle s_1, s_2\rangle, s_3\rangle, \alpha, \langle\langle s_1', s_2\rangle, s_3\rangle) \in\to_L$$

Since $\alpha \notin Act_2 \cup Act_3$ the respective states do not change. The handshaking of TS_2 and TS_3 does not have any impact on the usage of α. So there are also the transitions

$$(\langle s_1, \langle s_2, s_3\rangle\rangle, \alpha, \langle s_1', \langle s_2, s_3\rangle\rangle) \in\to_R$$
$$\Leftrightarrow (f_\approx(\langle\langle s_1, s_2\rangle, s_3\rangle), \alpha, f_\approx(\langle\langle s_1', s_2\rangle, s_3\rangle)) \in\to_R$$

ii) $\alpha \in (Act_1 \cap Act_2) \setminus Act_3$

$\alpha \notin Act_3$ means, that we have to distinguish three further cases based on the current state $\langle\langle s_1, s_2\rangle s_3\rangle$:

1. $(s_1, \alpha, s_1') \in\to_1$ and $(s_2, \alpha, s_2') \in\to_2$:
   Since both current states s_1 and s_2 have an available α transition the handshaking of these two would create a state $\langle s_1, s_2\rangle \xrightarrow{\alpha} \langle s_1', s_2'\rangle$. Since $\alpha \notin Act_3$ the handshaking with TS_3 then only creates a crossproduct of all such $\langle s_1, s_2\rangle$ states with some state $s_3 \in S_3$. This would infer that

   $$(\langle\langle s_1, s_2\rangle s_3\rangle, \alpha, \langle\langle s_1', s_2'\rangle s_3\rangle) \in\to_L$$

   If we first compute the handshaking $TS_2||TS_3$ we would get states, where all transitions using α only change the state of TS_2. Performing the second handshake would synchronize the transitioning of TS_1 and TS_2 using α while still not changing s_3. So we get

   $$(\langle s_1, \langle s_2, s_3\rangle\rangle, \alpha, \langle s_1', \langle s_2', s_3\rangle\rangle) \in\to_R$$
   $$\Leftrightarrow (f_\approx(\langle\langle s_1, s_2\rangle, s_3\rangle), \alpha, f_\approx(\langle\langle s_1', s_2'\rangle, s_3\rangle)) \in\to_R$$

2. $(s_1, \alpha, s_1') \in\to_1$ and $(s_2, \alpha, s_2') \notin\to_2$:
   This would infer that we can not use any transition containing α, since both TS_1 and TS_2 would need to use an α transition. Since TS_2, which is in state s_2 has no possible transition for α, TS_1 is not allowed to move.
   So for such a state $\langle s_1, s_2\rangle \in (TS_1||TS_2)$ there exists no α transition. Also synchronising over TS_3 does not change anything about the α transitions. Therefore

   $$(\langle\langle s_1, s_2\rangle s_3\rangle, \alpha, \langle\langle s_1', s_2'\rangle s_3\rangle) \notin\to_L$$

   In order to fulfil the condition we now show that also the right side does not have such a transition.

Performing the handshake of $TS_2||TS_3$ would create the states $\langle s_2, s_3\rangle$. Since per assumption s_2 does not have a α transition also $\langle s_2, s_3\rangle$ does not have one. Now also synchronising TS_1 can not add a α transition, since it would have to synchronise with TS_2 and therefore s_2 must have an α transition. This is not the case. So

$$(\langle s_1, \langle s_2, s_3\rangle\rangle, \alpha, \langle s_1', \langle s_2', s_3\rangle\rangle) \notin \rightarrow_R$$

$$\Leftrightarrow (f_\approx(\langle\langle s_1, s_2\rangle, s_3\rangle), \alpha, f_\approx(\langle\langle s_1', s_2'\rangle, s_3\rangle)) \notin \rightarrow_R$$

3. $(s_1, \alpha, s_1') \notin \rightarrow_1$ and $(s_2, \alpha, s_2') \in \rightarrow_2$:
This case is completely analogue to the second case.

iii) $\alpha \in Act_1 \cap Act_2 \cap Act_3$

1. $(s_1, \alpha, s_1') \in \rightarrow_1$ and $(s_2, \alpha, s_2') \in \rightarrow_2$ and (s_3, α, s_3'):
First performing the $TS_1||TS_2$ handshake we receive states $\langle s_1, s_2\rangle$ with $\langle s_1, s_2\rangle \xrightarrow{\alpha} \langle s_1', s_2'\rangle$. Afterwards synchronizing with TS_3 means that every such mentioned state would also change s_3 to s_3', since they synchronize over α. So:

$$(\langle\langle s_1, s_2\rangle s_3\rangle, \alpha, \langle\langle s_1', s_2'\rangle s_3'\rangle) \in \rightarrow_L$$

Completely analogously we can first compute $TS_2||TS_3$ and then $TS_1||(TS_2||TS_3)$. This will yield the same result and thus:

$$(\langle s_1, \langle s_2, s_3\rangle\rangle, \alpha, \langle s_1', \langle s_2', s_3'\rangle\rangle) \in \rightarrow_R$$

$$\Leftrightarrow (f_\approx(\langle\langle s_1, s_2\rangle, s_3\rangle), \alpha, f_\approx(\langle\langle s_1', s_2'\rangle, s_3'\rangle)) \in \rightarrow_R$$

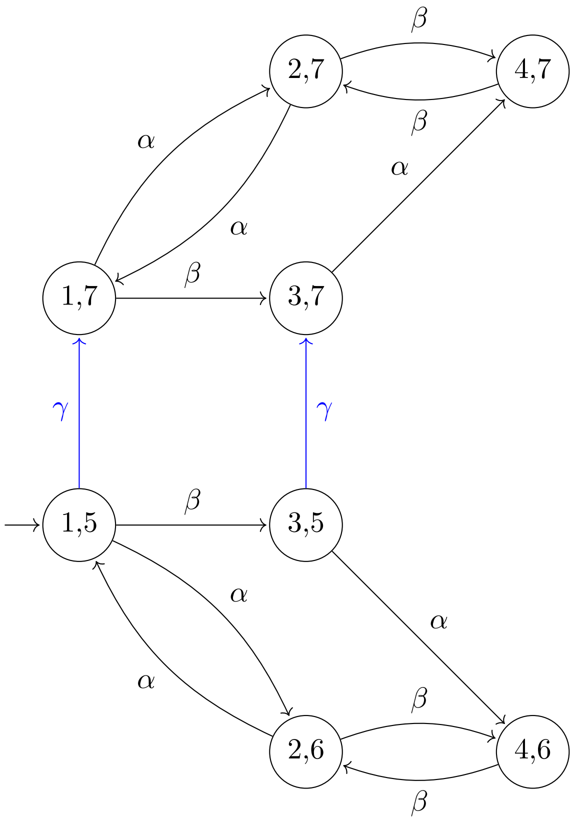
2. If any state $s_i \in \{s_1, s_2, s_3\}$ does not have an α transition we can, analogously to case ii.2) deduce, that there can not be an α transition for such a state $\langle\langle s_1, s_2\rangle s_3\rangle$ in $\rightarrow_L$ since this would need all s_i's to have an α transition.
Also we can directly infer that the same holds for $f_\approx(\langle\langle s_1, s_2\rangle s_3\rangle) = \langle s_1\langle s_2, s_3\rangle\rangle$ for the exact same reason.

This finally concludes in

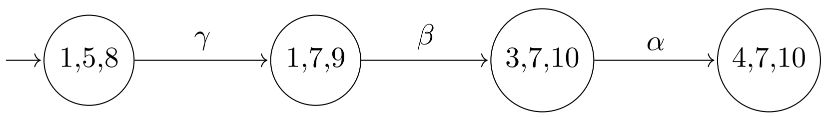
$$l) \xrightarrow{\alpha}_L (l') \Rightarrow f_\approx(l) \xrightarrow{\alpha}_R (l')$$

c)

First we build $TS_4 \coloneqq TS_1||TS_2$:



Now we have to build $TS_4||TS_3$:



Exercise 2

a)

The SOS-rules for LIFO channels with capacity 0 are the same as for FIFO channels with capacity 0.
The SOS-rules for LIFO communication, for a channel c with $cap(c) \geq 1$:

$$\frac{l_i \overset{c?x}{\hookrightarrow}_i l_i' \wedge \xi(c) = v_1 \dots v_{k-1}, v_k \wedge k \geq 1}{\langle l_1, \dots, l_i, \dots, l_n, \eta, \xi \rangle \xrightarrow{\tau} \langle l_1, \dots, l_i', \dots, l_n, \eta', \xi' \rangle}$$

$$\text{where } \eta' = \eta[x := v_k] \text{ and } \xi' = \xi[c \coloneqq v_1 \dots v_{k-1}]$$

for receiving and

$$\frac{l_i \overset{c!v}{\hookrightarrow}_i l_i' \wedge \xi(c) = v_1 \dots v_k \wedge k < cap(c)}{\langle l_1, \dots, l_i, \dots, l_n, \eta, \xi \rangle \xrightarrow{\tau} \langle l_1, \dots, l_i', \dots, l_n, \eta, \xi' \rangle}$$

$$\text{where } \xi' = \xi[c \coloneqq v_1 \dots v_k v]$$

for sending.

b)

Let M be a Turing machine, which simulates the LIFO channel system. Then we can construct a Turning machine M', which works like the following: simulate non-deterministically M and accept if M reaches state F.
Then M' accepts if F is reached at some point and rejects if F is not reached. So we can guarantee the reachability of F iff M' accepts. **Contradiction**
Reduction by subprogram technique:
Assuming there exists an algorithm A that decides the given problem. For a given Turing machine $T = (Q, \Sigma, \Gamma, \delta, q_0, F_T)$ the following LIFO channel system is created, which simulates the behavior of T:
There is one process P and two LIFO channels c_L and c_R. Both channels start and end at P, forming a loop. (If this is not allowed this can be simulated by an additional process and an additional channel with capacity 0 for each loop channel. P sends a synchronized message to the other process. The other process directly pushes the message on top of the LIFO stack.)
The states and the transitioning between states of P are the same as for the TM T. Moreover there is an additional accepting state F_P. Only reading to and writing from the tape has to be modeled differently:
The symbol inside the current cell is stored in a variable x. x initially has the value of the empty cell ($\square$), as the tape is empty in the beginning.
Whenever T moves left, the written new cell value is pushed to c_R, so that the tape content on the right side of T is stored in c_R. The top of c_L is popped and used as the new value of x (reading from left side). If c_L is empty, x is filled with the blank symbol $\square$. When T moves right, P behaves analogously, only left and right are switched.
If T halts (this can be checked for the current configuration as the tape symbol and current state are available), P moves to the accepting state F_P.
In F_P the channels are emptied and x is set to $\square$.
The algorithm A is now used to determine whether the state $F = (F_P, \eta_F, \xi_F)$ with $\eta_F(x) = \square$ and $\xi_F(c_L) = \xi_F(c_R) = \varepsilon$ can be reached.
Due to the construction above, this state can be reached iff T can reach a configuration in which it halts. Therefore, this solves the halting problem for Turing machines.

Exercise 3

Remarks:

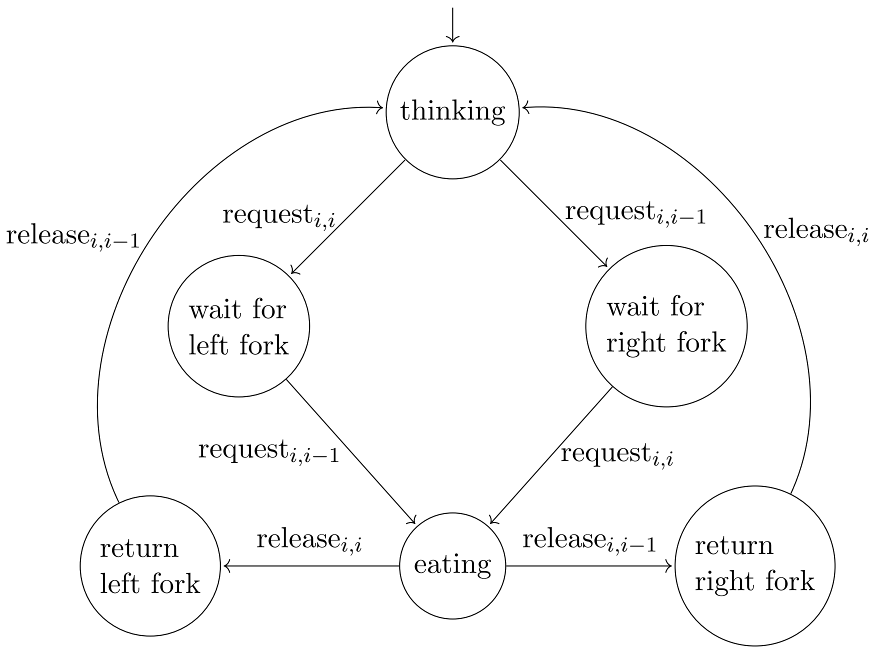
- In the following, calculations with i and j are in $\mathbb{F}_n$. For exercise b), where $n = 3$, this means that for $i = 2$ it holds $i + 1 = 2 + 1 = 3 \equiv_3 0$.
- We define the channel system $\mathcal{C}$ to have a channel capacity of 0 in order to model it as synchronous message passing. So in the following we don't write it as $c!x$ and $c?y$ to send and receive from a channel but model it as synchronous messages.
- Since no variables are used we omit the variable evaluation η.
  Since all channels have capacity 0 we omit the channel evaluation ξ.

a)

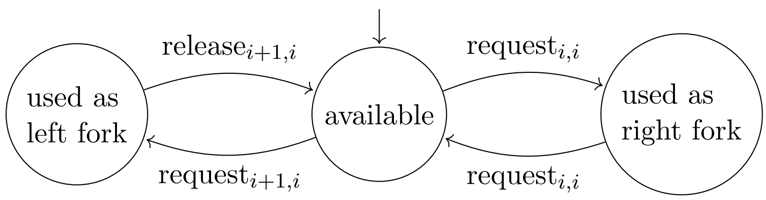
We model each philosopher by a program graph $\mathcal{P}_i$ and each fork by a program graph $\mathcal{F}_i$, $0 \leq i < n$. We use the following actions to synchronize the program graphs:

- $request_{i,j}$:
  Requesting the fork j based on the philosopher $\mathcal{P}_i$'s perspective ($j = i \Rightarrow$ right fork; $j = i-1 \Rightarrow$ left fork). This means we use the channel to the fork's program graph, which has a capacity of 0, to to synchronize two program graphs. Also note that we use i, j modulo n. So that the philosopher $\mathcal{P}_0$s left fork is the same as $\mathcal{P}_{n-1}$s right fork.
- $release_{i,j}$:
  Releasing the fork j which is currently used by philosopher $\mathcal{P}_i$ with the same scheme as above.

So the program graph $\mathcal{P}_i$ looks like the following:

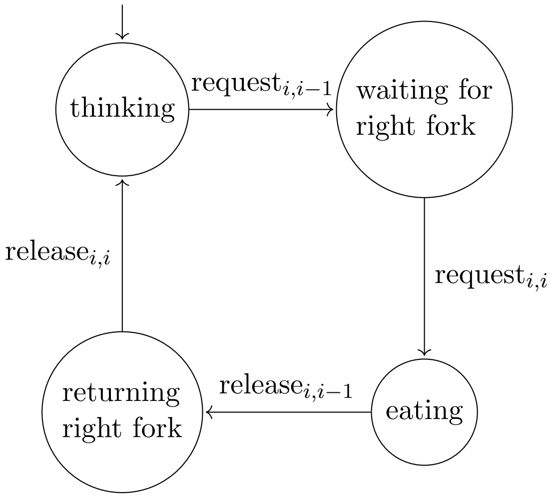


A fork $\mathcal{F}_i$ has the following program graph:

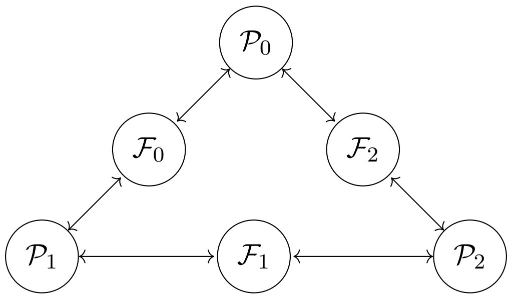


b)

The program graph can be simplified using the stated properties. The simplified program graph of philosopher $\mathcal{P}_i$ then follows with:



The program graphs $\mathcal{F}_i$ stay the same.
The complete channel system looks like the following:



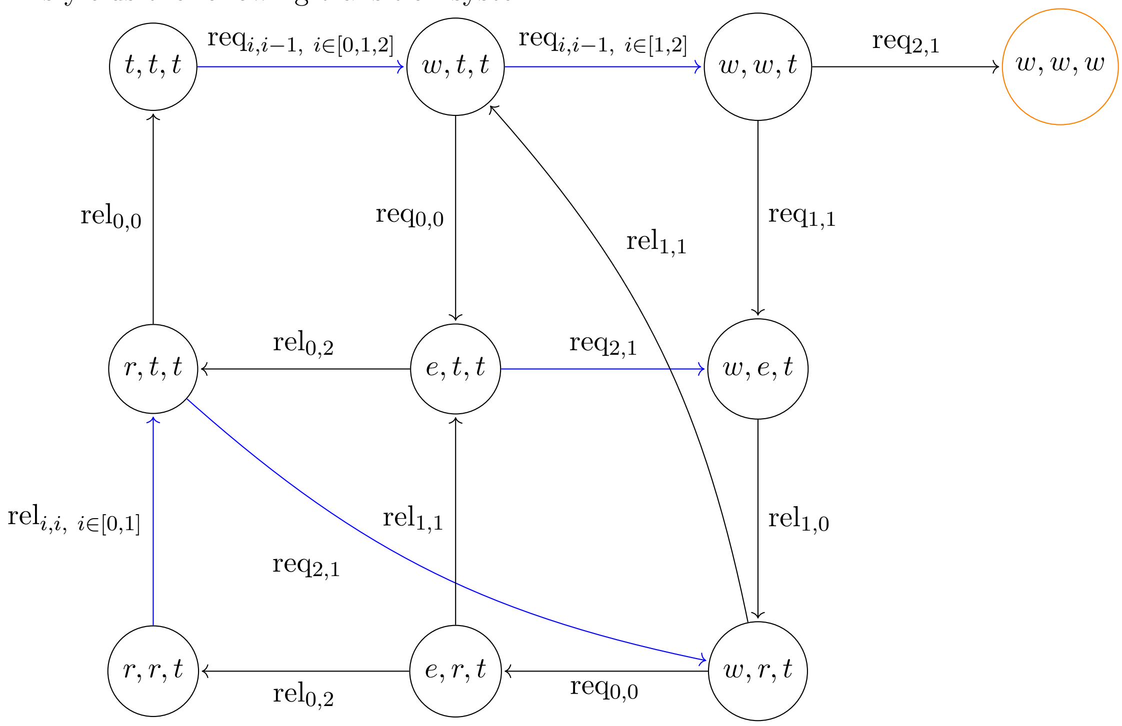
We can then construct the $TS(\mathcal{C}) = \mathcal{P}_0 \,||_H\, \mathcal{F}_0 \,||_H\, \mathcal{P}_1 \,||_H\, \mathcal{F}_1 \,||_H\, \mathcal{P}_2 \,||_H\, \mathcal{F}_2$ where $H = \{request_{i,j}, release_{i,j} \mid i,j \in [0,1,2], i = j \vee j = i-1\}$

We use the following simplifications to construct the transition system:

- We use $t, w, e,$ and r as a shorthand for the states thinking, waiting for right fork, eating, and returning left fork, respectively.
  We use req and rel as a shorthand for the actions request and release.
- The states of the Forks $\mathcal{F}_i$ can be derived from the states of the philosophers:
  $\mathcal{F}_i$ is in the state used as left fork iff $\mathcal{P}_{i+1}$ is in the state w or e.
  $\mathcal{F}_i$ is in the state used as right fork iff $\mathcal{P}_i$ is in the state e or r.
  $\mathcal{F}_i$ is in the state available otherwise.
- These simplifications lead to triples as states, where the i-th entry represents the state of philosopher $\mathcal{P}_i$.

- The rotational symmetry of the system allows us to consider the following states as equal under isomorphisms:
  (x, y, z), (y, z, x), and (z, x, y) for $x, y, z \in \{t, w, e, r\}$.
  This is used by blue colored edges.

This yields the following transition system:



c)

Yes. If every philosopher would block his/her left fork and wait for the right fork to be freed in order to get into the *eating* phase we observe a deadlock. This state is marked in orange.