Intuition to the construction from last time:

The construction makes heavy use
of the following trick in automata theory:

to shuffle two languages $L(A_1)$ and $L(A_2)$
over disjoint alphabets $\Sigma_1 \cap \Sigma_2 = \emptyset$,
extend A_1 by loops Σ_2 on each state,

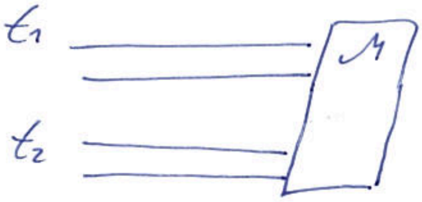
and A_2 by loops Σ_1.
The resulting automata A_1' and A_2' satisfy

$$\mathcal{L}(A_1') \cap \mathcal{L}(A_2') = \mathcal{L}(A_1) \sqcup\!\sqcup \mathcal{L}(A_2).$$

Example:

$\{ab\} \sqcup\!\sqcup \{xyz\}$

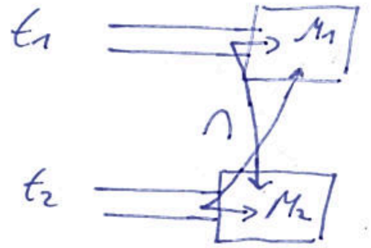
1.) The memory sees a shuffle
of the stores from both threads:



For example, the memory may see

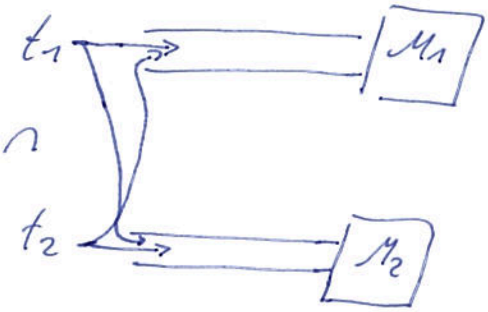
$$\rho = st(a, 1, t_1).\ st(b, 2, t_2).\ st(a, 2, t_1).$$

2.) Using the above idea,
the shuffle can be obtained
by synchronizing (intersection) the two automata $t_1 = \boxed{M_1}$
and $t_2 = \boxed{M_2}$ on the memory updates:



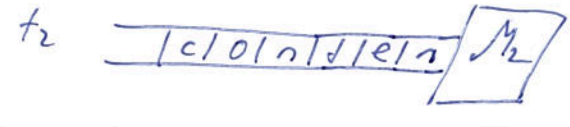
Here, one really defines alphabets (Σ_i consists of stores $st(a, v, t_i)$, $i = 1, 2$)
and applies the loop trick.

3.) Since the channels are FIFO,
the stores leave the buffer in the order
in which they are put into the buffer.
So instead of guessing the buffered stores of t_2
<u>at the memory</u>,
thread t_1 guesses the stores of t_2
already <u>when issuing</u> commands into the buffer:

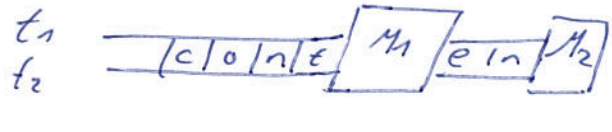


4.) Now the content of both buffers is identical,
up to the fact that M_1 and M_2
process the buffer at different speed:

and



can be understood as one buffer
with two pointers:

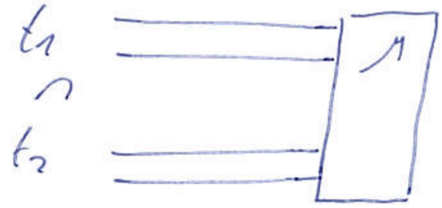


Remark:

One may think that it should be sufficient to guess the sequence of memory updates

$$\rho = sd(a,1,t_1).\,sd(b,2,t_2).\,sd(a,2,t_1)$$

and write this shared sequence into both buffers:



This, however, yields a <u>strict under-approximation</u> of TSO.

TSO requires that sequence ρ results from store actions <u>leaving the buffer</u>.

The above model <u>additionally</u> enforces this order when the stores are <u>send to</u> the buffer.

↳ This is more than TSO asks for.

↳ That the two memories can process the shared buffer at different speed fixes the problem.