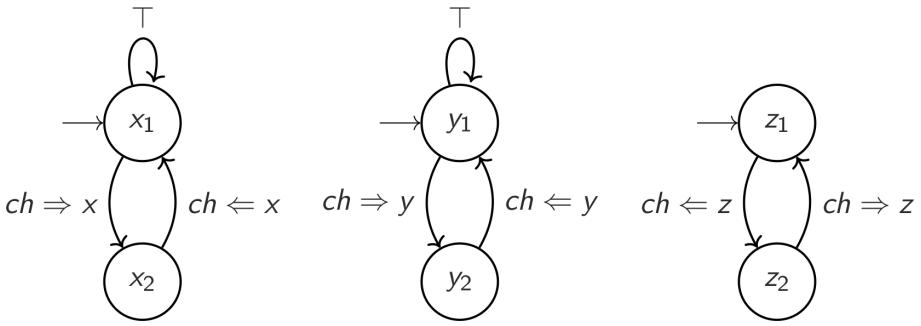
Question 1 *Reasoning about message passing[5 marks]*

Here is a critical section algorithm that uses synchronous message passing:

Algorithm 2.1: Mutual exclusion with server		
channel ch of bool		
x	**y**	**z**
forever do	forever do	forever do
x1: ch ⇒ x	y1: ch ⇒ y	z1: ch ⇐ z
x2: ch ⇐ x	y2: ch ⇐ y	z2: ch ⇒ z

Only processes x and y are competing for the critical section; z is an auxiliary process. The critical sections are at lines $x2$ and $y2$; x_1 and y_1 are the non-critical sections. The program variables x, y, z are just dummies; their values and types are unimportant.

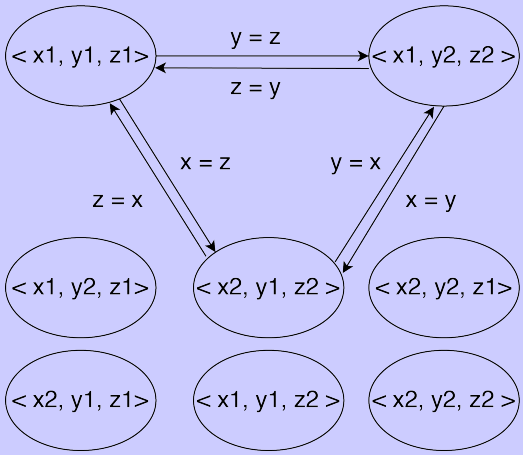
The transition diagrams for these processes are as follows.



(The self-loop is not depicted in the code above; it represents the ability to stay in the non-critical section forever).

1.1 Construct the closed product of these transition diagrams. The initial state will be $\langle x_1, y_1, z_1 \rangle$.

Answer:



1.2 It's obvious from inspection of the closed product that this algorithm satisfies mutual exclusion. Why?

Answer:

- When x_1 is entering in to critical section (ch $\Rightarrow$ x), $they_1$ and z_2 cannot entering.
- When y_1 is entering in to critical section (ch $\Rightarrow$ x), $thex_1$ and z_2 cannot entering.
- When z_2 is entering in to critical section (ch $\Rightarrow$ x), $thex_1$ and y_1 cannot entering.

No two processes are in their critical section at the same time. Therefore, it is mutual exclusion.

1.3 Does this algorithm satisfy eventual entry? Briefly motivate.

Answer:

I don't think so. This may be weak fairness, because if y_1 wants to enter the critical section, but it is always preempted by x_1 or z_2, y_1 cannot entry eventual.

1.4 Does this algorithm still work if we flip all inputs to outputs, and vice versa? Brifely motivate.

Answer:

No, it's a synchronous channel, when the channel has only input or only output, the input and output are meaningless, because all the input values to the channel and output values from the channel cannot be performed, so the whole algorithm is invalid.

1.5 The algorithm behaves oddly if we make ch asynchronous. Describe a scenario that (a) assumes an asynchronous, reliable channel; (b) goes on forever in a cycle; and (c) takes transitions other than the self-loops at x_1 and y_1 infinitely often; and (d) never visits locations x_2 and y_2.

Answer:

When the channel is asynchronous, input and output are not required at the same time.

Therefore, when z_1 input does not proceed, x_1 and y_1 cannot enter the critical section because there is no matching content in the channel. After z_1 input, z_2 enters the critical section before x_1 and y_1. z_1 input is used by z_2 output. When x_1 and y_1 access the channel again, there is no matching content in channel. When it go back to z_1 again, continue the same operation, it matches the situation of the topic.