a) Describe the flow of messages through the concurrent system, the possible synchronisations, and the possible sequences of messages. Identify any problems. [15%]

The *Keyboard* process outputs *key(x)* to *Plugboard*. *Plugboard* receives this on its right channel (r) and outputs the mapped letter (the result of *fplug(x)*) on its left channel (l). Practically speaking, this is passed through three *Rotors*. In the context of CCS, each *Rotor* synchronises with the input that comes in on r and outputs a transformed value on l for the next (or, in the case of the final *Rotor*, the *Reflector*) to synchronise with.

The *Reflector* synchronises with the input to its input channel (*in*) and broadcasts an output on its output channel (*out*). This output is the original input from *in* transformed by the reflector function *frefl(x)*. The input is then synchronised with by each *Rotor* on their right channel and output, transformed by *frotor(p,x)*, for the next *Rotor* (or, in the case of the final *Rotor*, the *Plugboard*) to synchronise with. The

I identified the following problems:

- After $Plugboard$ has output the result of $f_{plug}(x)$ to whichever channel it should use, it can still synchronise with messages on the same side of the board. This should be resolved by accepting from R first, then accepting from
    L. Practically, this means that if multiple $Keyboard$s were used with $Enigma$, more characters could be typed on other $Keyboard$s and sent through the $Plugboard$.
- *Rotor*s have a similar issue that would allow the characters to continue through $Enigma$ to the point of displaying the character. In this instance, there could also be race conditions between incrementing the rotors and sending the encrypted character. These are made void by the fact that $Keyboard$ cannot synchronise with inputs until the $\overline{inc}$ signal is received through the plugboard - that is, any transmissions on the $Rotors$ have already occurred.
- After a *Rotor* has synchronised with l or r once, it will not synchronise with inc_r again. It enters a recursion on $RotorFunction(p)$, which cannot at any point synchronise with inc_r. What this means practically is that rotors, in this implementation, can be incremented only once.

b) Modify the model to make it a more accurate representation of the real Enigma system, and explain briefly how your version overcomes the problems you identified in (a).

$RotorFunction$ should not be recursive or provide a choice. It should receive from the right, then the left, to ensure that data flows both ways without an increment happening during the flow of data through the machine:

$$\begin{aligned} RotorFunction(p) &= r(x).\bar{l}(f_{rotor}(p,x)).\\ &\ l(x).\bar{r}(f_{rotor}(p,x)) \end{aligned}$$

Instead, choosing $RotorFunction$ should follow with a recursion on $Rotor$. This means that $Rotor$ responds to inc_r when data is not flowing through the machine.

$$\begin{aligned} Rotor(26,p) &= inc_r.\overline{inc_l}.Rotor(0,p-26) + RotorFunction(p).Rotor(26,p)\\ Rotor(c,p) &= inc_r.Rotor(c+1,p+1) + RotorFunction(p).Rotor(c,p) \end{aligned}$$

$Plugboard$ should only respond from the right first, similarly:

$$\begin{aligned} Plugboard &= r(x).\bar{l}(f_{plug}(x)).\\ &\ l(x).\bar{r}(f_{plug}(x)).Plugboard \end{aligned}$$

$$\begin{array}{lcl} Reflector & = & in(x).\overline{out}(f_{refl}(x)).Reflector \\ \\ Keyboard & = & \overline{key}(x).\overline{inc}.lamp(y).Keyboard \\ \\ Plugboard & = & r(x).\bar{l}(f_{plug}(x)). \\ & & l(x).\bar{r}(f_{plug}(x)).Plugboard \\ \\ Rotor(26,p) & = & inc_r.\overline{inc_l}.Rotor(0,p-26) + RotorFunction(p).Rotor(0,p-26) \\ Rotor(c,p) & = & inc_r.Rotor(c+1,p+1) + RotorFunction(p).Rotor(c,p) \\ RotorFunction(p) & = & l(x).\bar{r}(f_{rotor}(p,x)) \\ & + & r(x).\bar{l}(\overline{f_{rotor}}(p,x)) \\ \\ Enigma & = & Reflector[ref/in, ref/out] \\ & | & Rotor(c_3,p_3)[ref/l, m1/r, i3/inc_r] \\ & | & Rotor(c_2,p_2)[m1/l, m2/r, i3/inc_l, i2/inc_r] \\ & | & Rotor(c_1,p_1)[m2/l, m3/r, i2/inc_l, i1/inc_r] \\ & | & Plugboard[m3/l, keys/r] \\ & | & Keyboard[keys/key, keys/lamp, i1/inc] \end{array}$$

Figure 1: My modification to the abstract CCS/Pi-calculus model of the Enigma system components