The Assessment of Distributed System Part 1

Letian Wang
lwang73@sheffield.ac.uk

May 2018

1 Question a

Describe the flow of messages through the concurrent system, the possible synchronisations, and the possible sequences of messages. Identify any problems.

1.1 The model details of Figure 4

I think the concurrent system give a basic model of the Enigma system. It shows the function of every component in Enigma. However, it cannot simulate the Enigma machine in the real world.

In this paragraph, I will give some details about the flow of messages in the concurrent system of Figure 4. The Enigma machine is simulated by six processes working parallel including Keyboard, Plugboard, three Rotors and a Reflector.

First of all, the original message will be given from the Keyboard. When a key is pressed, it will send both a signal of key x on the key channel, which is renamed by keys after this process and an increment signal inc which will make the rotor move. Besides, the lamp channel is also renamed by keys which may lead a problem that the output and input in channel keys may do a τ (problem 1). Although there is an input of channel keys in process Plugboard before this input in Keyboard in the real Erlang program, it may lead to a problem in the model of Figure 4. The model of Keyboard, after renaming, will be:

$$Keyboard = \overline{keys}\langle x\rangle.\overline{i1}.keys(y).Keyboard \tag{1}$$

.

Secondly, the message of inc and key x need to go through the Plugboard to the Rotors. Plugboard process will rename r channel into keys and wait for a message from keys channel from process Keyboard. After that, a plug function makes the message be encrypted by an external wire and the pressed key character will be translated into another one. The message will output by the channel l and the RotorFunction will receive this message. The model of Plugboard, after renaming, will be:

$$\begin{aligned} Plugboard = \quad & keys(x).\overline{m3}\langle f_{plug}(x)\rangle.Plugboard \\ + \quad & m3(x).\overline{keys}\langle f_{plug}(x)\rangle.Plugboard \end{aligned} \tag{2}$$

.

Thirdly, the message is transferred into three Rotors. It waits for an increment signal which is sent by Keyboard. However, there is another problem which is different from the physical environment. Moreover, when the Rotor has been incremented 26 times, it will give an increment signal to its left-hand neighbour. The message would be encrypted at this stage through the position of Rotor. When RotorFunction receives the message from l channel, it encrypts the message from the position of Rotor. After that, it output the message by r channel to the next Rotor. In the last Rotor before Reflector, the message will send the message to Reflector by a ref channel which renames the l channel without the increment signal.

Importantly, because the physical machine transfer the increment signal before the encode, it means that in the Rotor model, after the $inc_r.Rotor(c+1, p+1)$ process the equation only left the $RotorFunction(p)$ function to wait the encode process.

The model of Rotor, after renaming , will be:
the first Rotor:

$$Rotor(26, p_1) = i1.\overline{i2}.Rotor(0, p_1 - 26) + RotorFunction(p_1) \tag{3}$$

,

$$Rotor(c_1, p_1) = i1.Rotor(c_1 + 1, p_1 + 1) + RotorFunction(p_1) \tag{4}$$

,

$$\begin{aligned} RotorFunction(p_1) = \quad & m2(x).\overline{m3}\langle f_{rotor}(p_1, x)\rangle.RotorFunction(p_1) \\ + \quad & m3(x).\overline{m2}\langle \overline{f_{rotor}}(p_1, x)\rangle.RotorFunction(p_1) \end{aligned} \tag{5}$$

;
the second Rotor:

$$Rotor(26, p_2) = i2.\overline{i3}.Rotor(0, p_2 - 26) + RotorFunction(p_2) \tag{6}$$

,

$$Rotor(c_2, p_2) = i2.Rotor(c_2 + 1, p_2 + 1) + RotorFunction(p_2) \tag{7}$$

,

$$\begin{aligned} RotorFunction(p_2) = \quad & m1(x).\overline{m2}\langle f_{rotor}(p_2,x)\rangle.RotorFunction(p_2) \\ + \quad & m2(x).\overline{m1}\langle \overline{f_{rotor}}(p_2,x)\rangle.RotorFunction(p_2) \end{aligned} \tag{8}$$

;

the third Rotor:

$$Rotor(26,p_3) = i3.\overline{inc_l}.Rotor(0,p_3-26) + RotorFunction(p_3) \tag{9}$$

,

$$Rotor(c_3,p_3) = i3.Rotor(c_3+1,p_3+1) + RotorFunction(p_3) \tag{10}$$

,

$$\begin{aligned} RotorFunction(p_3) = \quad & ref(x).\overline{m1}\langle f_{rotor}(p_3,x)\rangle.RotorFunction(p_3) \\ + \quad & m1(x).\overline{ref}\langle \overline{f_{rotor}}(p_3,x)\rangle.RotorFunction(p_3) \end{aligned} \tag{11}$$

.

Fourthly, the message is transferred into the Reflector. It renames the in channel by ref channel and it can receive the message from the last Rotor. After a function of reflector, which is similar to the function in Plugboard, the character will be encrypted into another character. When it has been encrypted, it will send back by a ref channel which renames the channel out. The model of Keyboard, after renaming, will be:

$$Reflector = ref(x).\overline{ref}\langle f_{reft}(x)\rangle.Reflector \tag{12}$$

.

Fifthly, the message in the channel ref is received by the last Rotor. The message will be encrypted by the three Rotor again by the position of Rotor. Without any increment signal, the Rotor does not change position this time. The rightmost Rotor will use an m3 channel to output the encrypted message to the Plugboard.

Sixthly, the Plugboard receives the message from m3 channel which renames l channel. It encrypted it and sent it to the keys channel represent the r channel.

Finally, the encrypted message is received by the Keyboard keys channel representing the lamp channel. It will switch a light to the encrypted character.

1.2 Problem

1.2.1 Conflict in increment signal

In the physical system, the increment signal will make all Rotor change before the message transfer into Rotor. However, the model in Figure 4, the increment signal could transfer to the Rotor after the message return. The reason for this problem is that there is no restriction of the increment signal and the increment signal need send the message before channel keys.

1.2.2 Conflict in process between Keyboard and Plugboard

In the Keyboard process, when it sent a character by channel keys, both Keyboard process and Plugboard process has a channel keys waiting for the signal. This may lead to a problem that the message received by the Keyboard before the Plugboard. It means that the character may return to the lamp directly.

1.2.3 Confilict in Rotors

When all three Rotor is written separately, it could easily find they may lead to a potential deadlock. For example, when the first Rotor gets the signal in channel $m3$ from Plugboard, it needs to send a message in channel in $m2$. However, after it send a message by channel $m2$, it will repeat the $RotorFunction(p_1)$ of $Rotor(c_1, p_1)$ which is also wait a message from channel. It may lead to repeat by itself and do not transfer message to next Rotor.

1.2.4 An extra increment signal

There is an extra increment signal in the equation of Rotor CCS:

$$Rotor(26, p) = inc_r.\overline{inc_l}.Rotor(0, p - 26) + RotorFunction(p) \tag{13}$$

,

$$Rotor(c, p) = inc_r.Rotor(c + 1, p + 1) + RotorFunction(p) \tag{14}$$

.

It can be easily found that when value of c is 0-24, it has emerge 25 increment signals. However, when it is 25, it firstly will receive two increment signals from Eq. (13) and Eq.(14). It lead to a serious problem that there is 27 increment signals in a cycle.

1.2.5 RotorFunction

It is very confused that when it is going to run the RotorFunction, it will be trapped in the RotorFunction and never come back to Rotor. The equation is shown as:

$$\begin{aligned} RotorFunction(p) = \quad & l(x).\overline{r}\langle f_{plug}(p, x)\rangle.RotorFunction(p) \\ + \quad & r(x).\overline{l}\langle \overline{f_{plug}}(p, x)\rangle.RotorFunction(p) \end{aligned} \tag{15}$$

. In this equation, When then Rotor run the RotoFunction, it will always run RotorFunction again and agian. When a message has been sent back to keyboard, it needs to run Rotor again.

2 Question 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).

2.1 Solusion

2.1.1 Solution for increment signal

The solution need meet some requirements. The first requirement is the increment signal need send to the Rotors before the character message is sent to the Plugboard. The second requirement is that it needs a restriction in which process send or get increment.

2.1.2 Solution for process between Keyboard and Plugboard

The process in the physical system is the Keyboard process send a signal by channel keys, and the Plugboard process can get it. Thus, this process needs to add a restriction to Keyboard and Plugboard process. This restriction prevents Keyboard send the signal and get a lamp signal directly.

2.1.3 Solution for Rotors

The normal sending process in those Rotors is every Rotor get the message from its right-hand side and send it to the left-hand one(which exchange the right and left in return process). Hence, it needs some restrictions in those Rotor processes. Those restrictions could forbid they receive the message which is sent by themselves. This would prevent the deadlock for those Rotors.

2.1.4 Solution for an extra increment signal

In the 26^{th} Rotor function, the extra increment signal need to be deleted. Only in this way, it can prevent the extra increment.

2.1.5 Solution for RotorFunction

When the message has sent to the right one, the RotorFunction needs to restart a process Rotor. I think it need change both Rotor and RotorFunction.

3 Modified model

3.1 Modified process

In the real world, the increment signal needs to transfer to the Rotor before the message signal. Hence, the keyboard process needs to be modified. I think the process will be:

$$Keyboard = \overline{inc}.\overline{key}\langle x\rangle.lamp(y).Keyboard \tag{16}$$

. This sequence forces the Rotor change position before sending the character message.

For the extra increment, it will be modified as:

$$Rotor(26,p) = \overline{inc_l}.Rotor(0,p-26) + RotorFunction(p) \tag{17}$$

.

For the RotorFunction, it will be modified as:

$$Rotor(26,p) = \overline{inc_l}.Rotor(0,p-26) + RotorFunction(p).Rotor(26,p) \tag{18}$$

,

$$Rotor(c,p) = inc_r.Rotor(c+1,p+1) + RotorFunction(p).Rotor(c,p) \tag{19}$$

,

$$\begin{aligned} RotorFunction(p) = \quad & l(x).\bar{r}\langle f_{plug}(p,x)\rangle \\ + \quad & r(x).\bar{l}\langle \overline{f_{plug}}(p,x)\rangle \end{aligned} \tag{20}$$

. In this way, it always restart the Rotor equation and the RotorFunction can be seen only a function of Rotor.

Other equations in Fig. 4 is not changed, and they are:

$$\begin{aligned} Plugboard = \quad & r(x).\bar{l}\langle f_{plug}(x)\rangle.Plugboard \\ + \quad & l(x).\bar{r}\langle f_{plug}(x)\rangle.Plugboard' \end{aligned} \tag{21}$$

$$Reflector = in(x).\overline{out}\langle f_{reft}(x)\rangle.Reflector \tag{22}$$

.

3.2 Modified Enigma system

After the modified, the $CCS/\pi - calculus$ model of Enigma system has been added some restriction. It will be:

$$\begin{aligned}
Enigma = \quad & (new \quad ref)Reflector[ref/in, ref/out] \\
| \quad & (new \quad i3, m1, ref)Rotor(c_3, p_3)[ref/l, m1/r, i3/inc_r] \\
| \quad & (new \quad i2, i3, m1, m2)Rotor(c_2, p_2)[m1/l, m2/r, i3/inc_l, i2/inc_r] \\
| \quad & (new \quad i1, i2, m2, m3)Rotor(c_1, p_1)[m2/l, m3/r, i2/inc_l, i1/inc_r] \\
| \quad & (new \quad keys, m3)Plugboard[m3/l, keys/r] \\
| \quad & (new \quad keys, i1)Keyboard[keys/key, keys/lamp, i1, inc]
\end{aligned} \tag{23}$$

. In the modified model, I try to give some restriction to prevent the process sending and getting a message to themselves. I force them to pass the message to another process which is waiting for the signal on the same channel. Those restrictions make them have to do a τ. For example, in the Keyboard and Plugboard processes, When there is no restriction, the keys channel in Keyboard process could receive a lamp signal in keys channel which is sent by itself. The restriction ensures that they have to do a τ following the basic rule and the message could transfer between those processes.