Erl-nigma

Jack Cheng Ding Han

Module: COM3190 Theory of Distributed Systems

Department of Computer Science
University of Sheffield

May 2018

Contents

1 Model 1 **1**

2 Model 2 **3**

References **5**

List of Figures

1.1 The original model provided by the assignment specification [1]. . . 1

2.1 New model. 3

Chapter 1

Model 1

(a) Describe the flow of messages through the concurrent system, the possible synchronisations, and the possible sequences of messages. Identify any problems [1].

$$
\begin{aligned}
\text{Reflector} =& \text{in}(x).\overline{\text{out}}\langle f_{\text{refl}}(x)\rangle.\text{Reflector} \\
\text{Keyboard} =& \overline{\text{key}}\langle x\rangle.\overline{\text{inc}}.\text{lamp}(y).\text{Keyboard} \\
\text{Plugboard} =& r(x)).\bar{l}\langle f_{\text{plug}}(x)\rangle.\text{Plugboard} \\
&+ l(x)).\bar{r}\langle f_{\text{plug}}(x)\rangle.\text{Plugboard} \\
\text{Rotor}(26,p) =& \text{inc}_{\text{r}}.\overline{\text{inc}_{\text{l}}}.\text{Rotor}(0,p-26) + \text{RotorFunction}(p) \\
\text{Rotor}(c,p) =& \text{inc}_{\text{r}}.\text{Rotor}(c+1,p+1) + \text{RotorFunction}(p) \\
\text{RotorFunction}(p) =& l(x).\bar{r}\langle f_{\text{rotor}}\,(p,x)\rangle.\text{RotorFunction}(p) \\
&+ r(x).\bar{l}\left\langle \overline{f_{\text{rotor}}}(p,x)\right\rangle.\text{RotorFunction}(p) \\
\text{Enigma} =& \text{Reflector}\,[\text{ref/in},\text{ref/out}] \\
&|\text{Rotor}(c_3,p_3)\,[\text{ref/l},\text{m1/r},\text{i3/inc}_r] \\
&|\text{Rotor}(c_2,p_2)\,[\text{m1/l},\text{m2/r},\text{i3/inc}_l,\text{i2/inc}_r] \\
&|\text{Rotor}(c_3,p_3)\,[\text{m2/l},\text{m3/r},\text{i2/inc}_l,\text{i1/inc}_r] \\
&|\text{Playboard}\,[\text{m3/l},\text{keys/r}] \\
&|\text{Keyboard}\,[\text{keys/key},\text{keys/lamp},\text{i1/inc}]
\end{aligned}
$$

Figure 1.1: The original model provided by the assignment specification [1].

Suppose all the increment counts and positions are 0 and the first character in the message a is typed in.

Keyboard sends a over the channel keys and then it sends a signal over the channel i1 to increment first Rotor. Reduce twice using the $\bar{a}\langle y\rangle.P$

rule.

$$\text{Keyboard} \xrightarrow{\overline{\text{keys}}\langle a \rangle} \text{Keyboard}'$$
$$\text{Keyboard}' \xrightarrow{\overline{\text{i1}}} \text{Keyboard}''$$

A possible sequence to follow from this has the channel keys synchronised as Keyboard is sending over the channel keys and Plugboard is receiving from it. Plugboard can only receive on m3 after a is encrypted by the three Rotors and the Reflector. So reducing with the $P + Q$ rule, Plugboard must receive a over the channel keys and bind the result to x (the $\overline{x}(y).P$ rule). Which means it must then send the output of over the channel m3.

$$\text{Plugboard} \xrightarrow{\text{keys}(a)} \text{Plugboard}'$$

A problem arises with the Rotors if they do not receive signals in the correct order. The first Rotor has a choice between receiving the increment signal on channel i1 and advancing itself or performing RotorFunction. RotorFunction would receive the symbol on channel m3 and send the output of f_{rotor} at the current position when the symbol is inputted. In an actual Enigma machine, the rotors should increment before they encrypt but here, x could be received over channel m3 before i1 is received. If that happens, it cannot continue to behave like Rotor (i.e. ignoring the recursive call), and instead continues to behave like RotorFunction because of the reduction rule $P + Q$.

$$\text{Rotor}(0,0) \xrightarrow{\text{m3}(a)} \bar{l}\left\langle \overline{f_{\text{rotor}}}(0,a) \right\rangle.\text{RotorFunction}(0) \tag{1.1}$$

Chapter 2

Model 2

(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). [1].

$$
\begin{aligned}
\text{Reflector} =& \text{in}(x).\overline{\text{out}}\langle f_{\text{refl}}(x)\rangle.\text{Reflector} \\
\text{Keyboard} =& \overline{\text{inc}}.\overline{\text{key}}\langle x\rangle.\text{lamp}(y).\text{Keyboard} \\
\text{Plugboard} =& r(x)).\bar{l}\langle f_{\text{plug}}(x)\rangle.l(x)).\bar{r}\langle f_{\text{plug}}(x)\rangle.\text{Plugboard} \\
\text{Rotor}(26,p) =& \text{inc}_\text{r}(1).R(r,p+1,x).r(x) \\
& .R(l,p+1,x).\text{Rotor}(0,p-26) \\
& + \text{inc}_\text{r}(0).\overline{\text{inc}_\text{l}}(0).l(x).R(r,p,x).r(x) \\
& .R(l,p,x).\text{Rotor}(26,p) \\
\text{Rotor}(c,p) =& \text{inc}_\text{r}(1).\overline{\text{inc}_\text{l}}(0).l(x).R(r,p+1,x).r(x) \\
& .R(l,p+1,x).\text{Rotor}(c+1,p+1) \\
& + \text{inc}_\text{r}(0).\overline{\text{inc}_\text{l}}(0).l(x).R(r,p,x).r(x) \\
& .R(l,p,x).\text{Rotor}(c,p) \\
R(d,q,x) =& \overline{d}\langle f_{\text{rotor}}\,(q,x)\rangle \\
\text{Enigma} =& \text{Reflector}\,[\text{ref/in},\text{ref/out}] \\
& |\text{Rotor}(c_3,p_3)\,[\text{ref/l},\text{m1/r},\text{i3/inc}_r] \\
& |\text{Rotor}(c_2,p_2)\,[\text{m1/l},\text{m2/r},\text{i3/inc}_l,\text{i2/inc}_r] \\
& |\text{Rotor}(c_3,p_3)\,[\text{m2/l},\text{m3/r},\text{i2/inc}_l,\text{i1/inc}_r] \\
& |\text{Playboard}\,[\text{m3/l},\text{keys/r}] \\
& |\text{Keyboard}\,[\text{keys/key},\text{keys/lamp},\text{i1/inc}]
\end{aligned}
$$

Figure 2.1: New model.

Closer to a real Enigma machine, the Keyboard sends the increment signal before the symbol to be encrypted unlike the model in Fig. 1.1.

A Rotor process can continue to behave like the Rotor process in two choices where value received on the increment channels is 1. To prevent deadlock like the model in Fig. 1.1, when a symbol to be encrypted is received on the channel l earlier than the signal to increment the rotor, the process must receive a value of 0 over the increment channels.

References

[1] R. G. Taylor, *COM3190/COM6116 programming assignment Erl-nigma – a simulated Enigma machine*, May 2018.