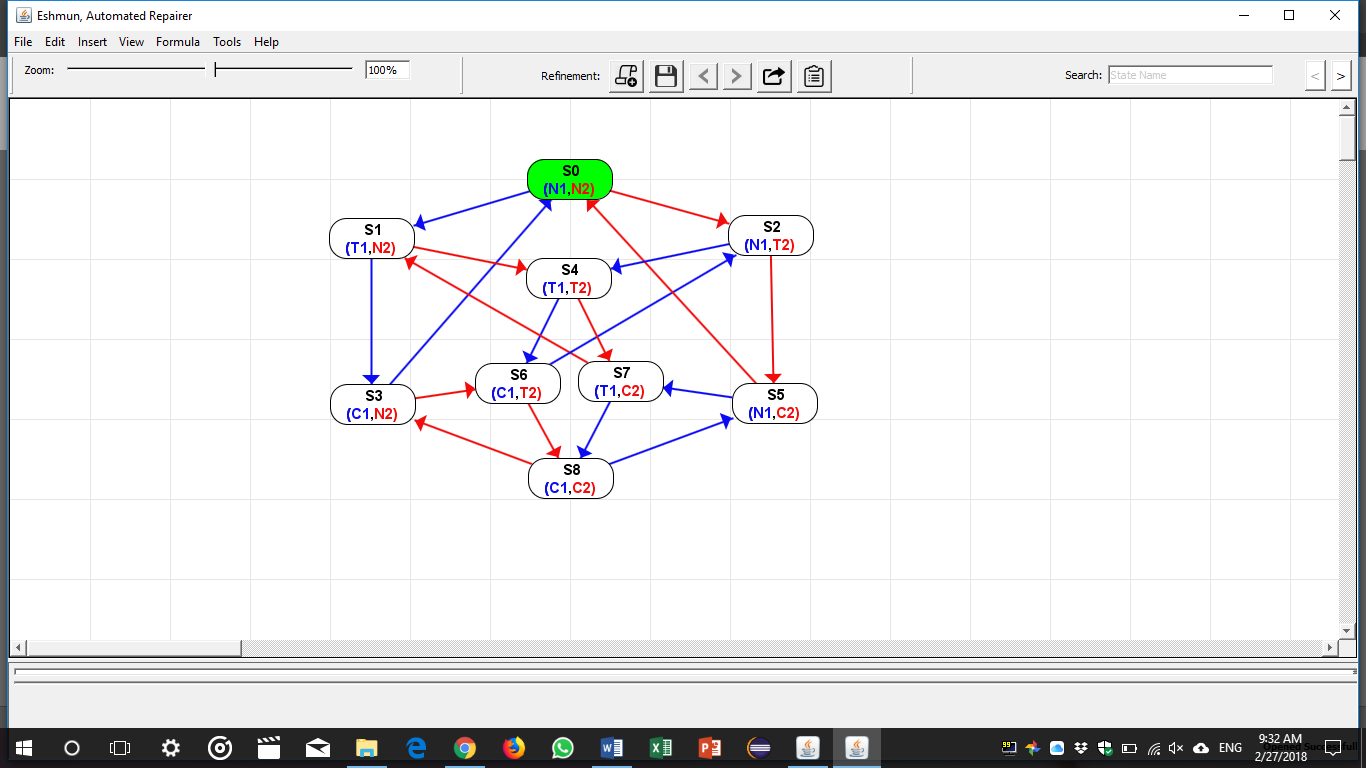
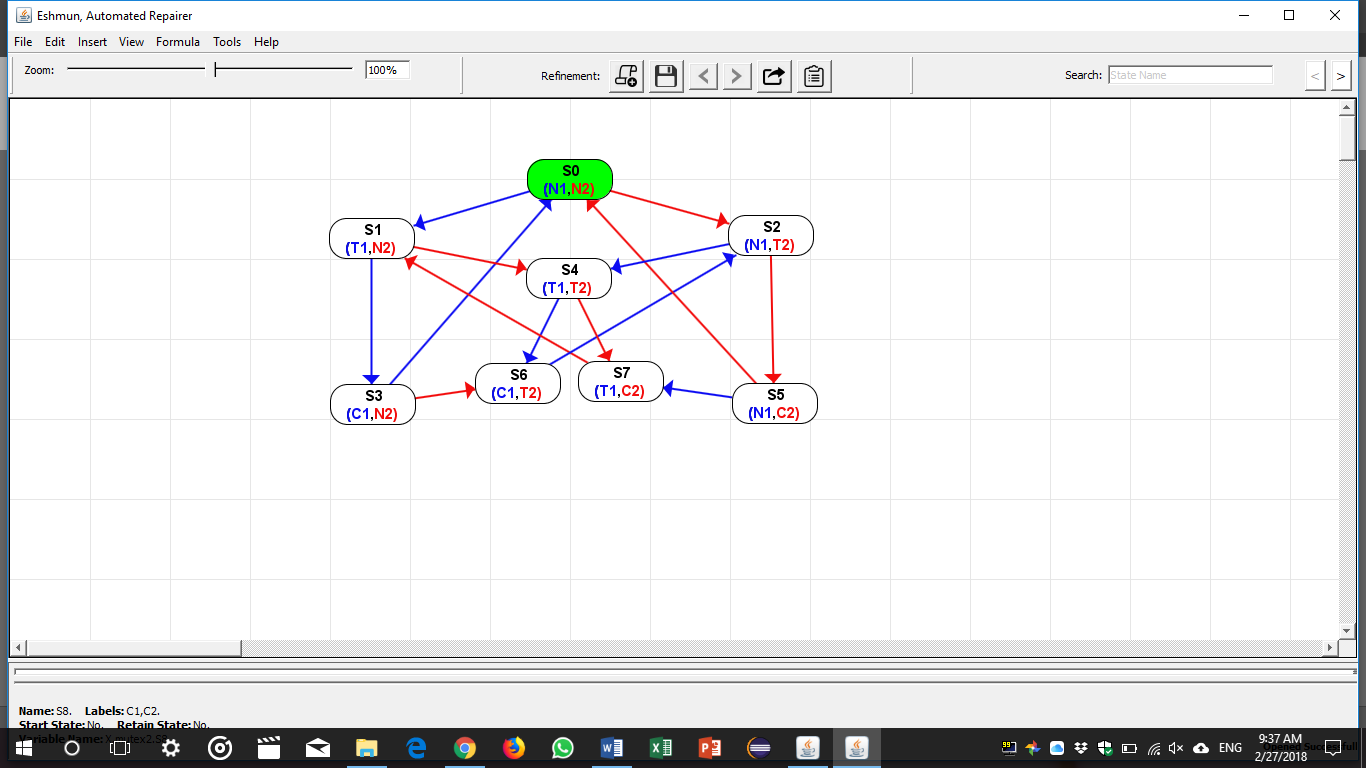
**Problem 1:**

**Solving Mutex Liveness Problem (Procedure):**

1. First, I reconstructed the Mutex model without the liveness Property. This model has 9 states (32) since every process has 3 different states and we have 2 processes.



1. I removed state S8 since it is an illegal state.



1. From there onward, no more states can be removed. Only new states can be added and the existing states can be modified. This implies that some of the transitions will be modified as well.
2. In order to make the mutex live, we should guarantee equal chance for both processes to enter the critical section. To achieve that, a process that enters the critical section should be forced to allow the other process to enter the critical section.

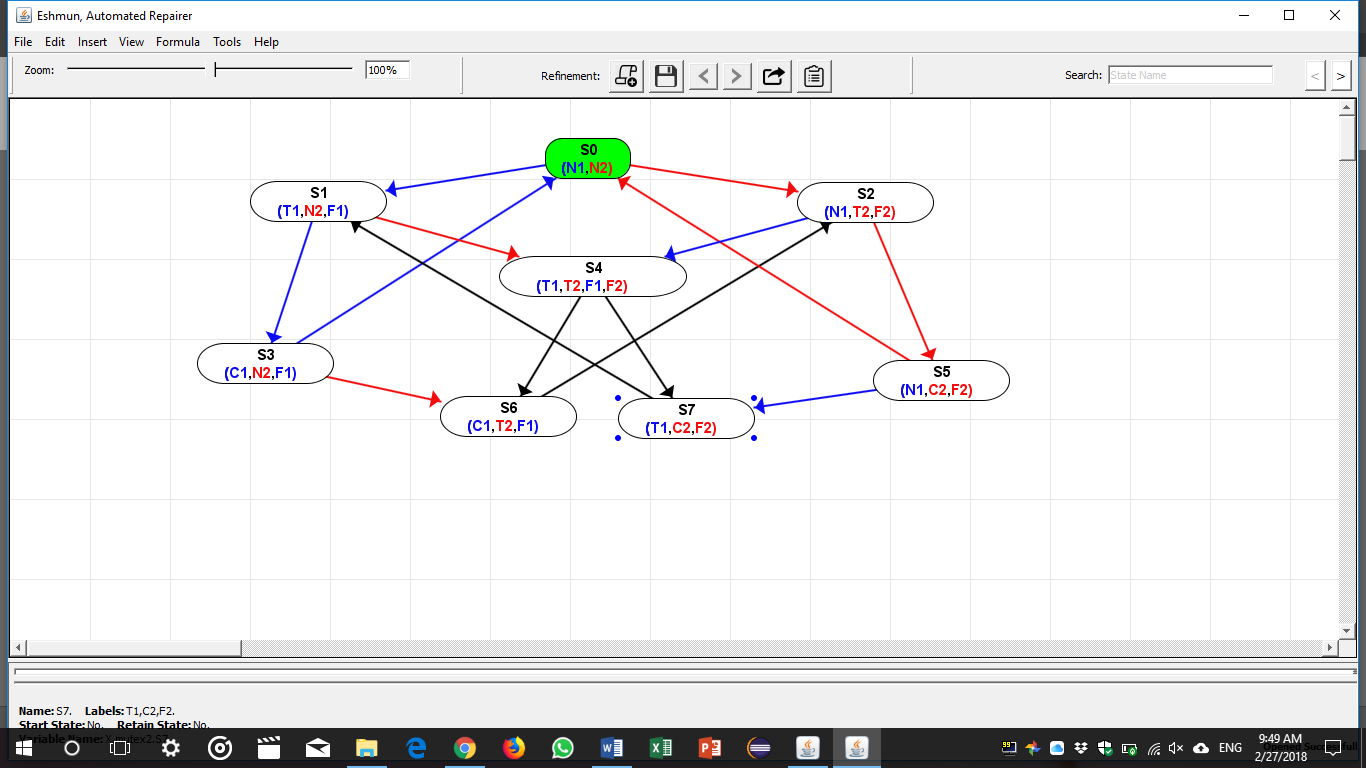
To do this, we will need a variable F (consider it as a lock) so that you cannot enter the critical section unless you have your lock.

Let F1 denote the lock that process 1 need to have to enter the CS.

Let F2 denote the lock that process 2 need to have to enter the CS.

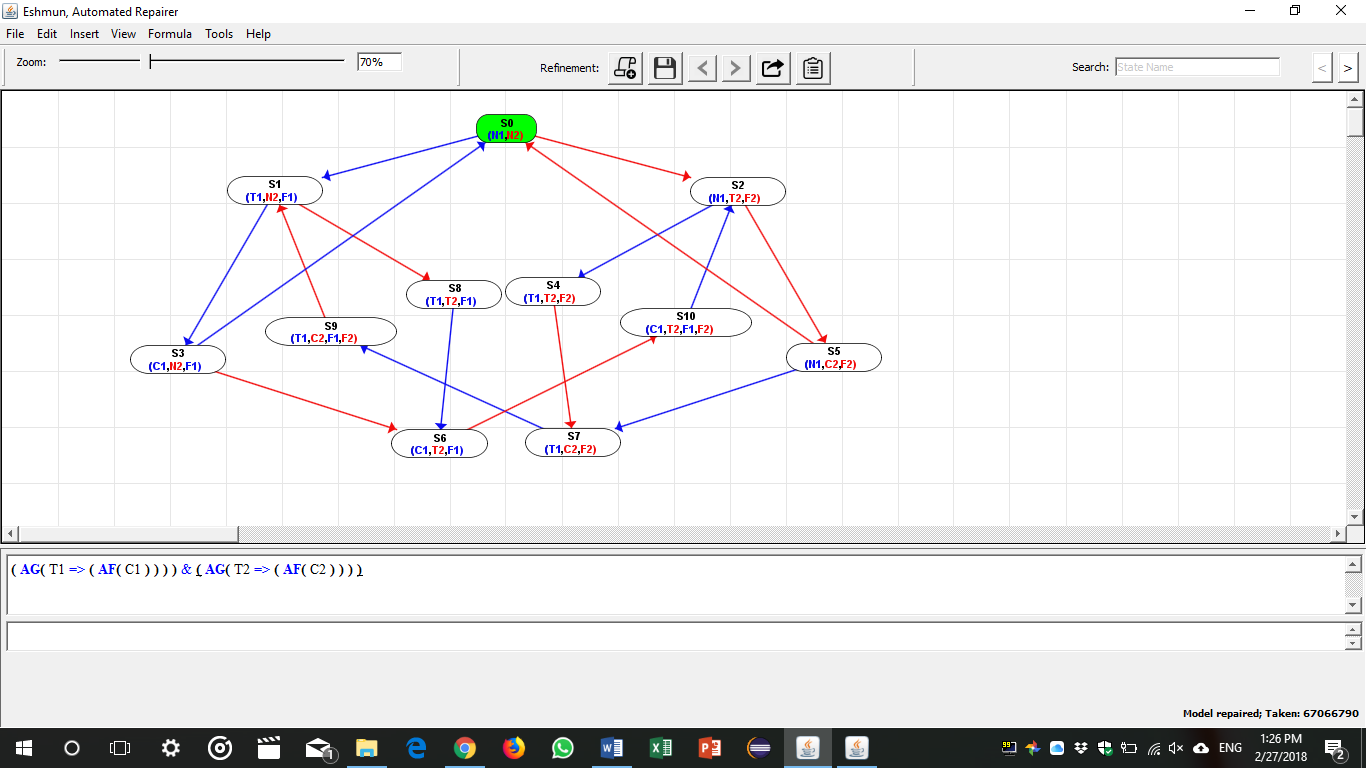
Observation: We might need up the 16 states (42) since every process now will have 4 different states (N, T, C, F)

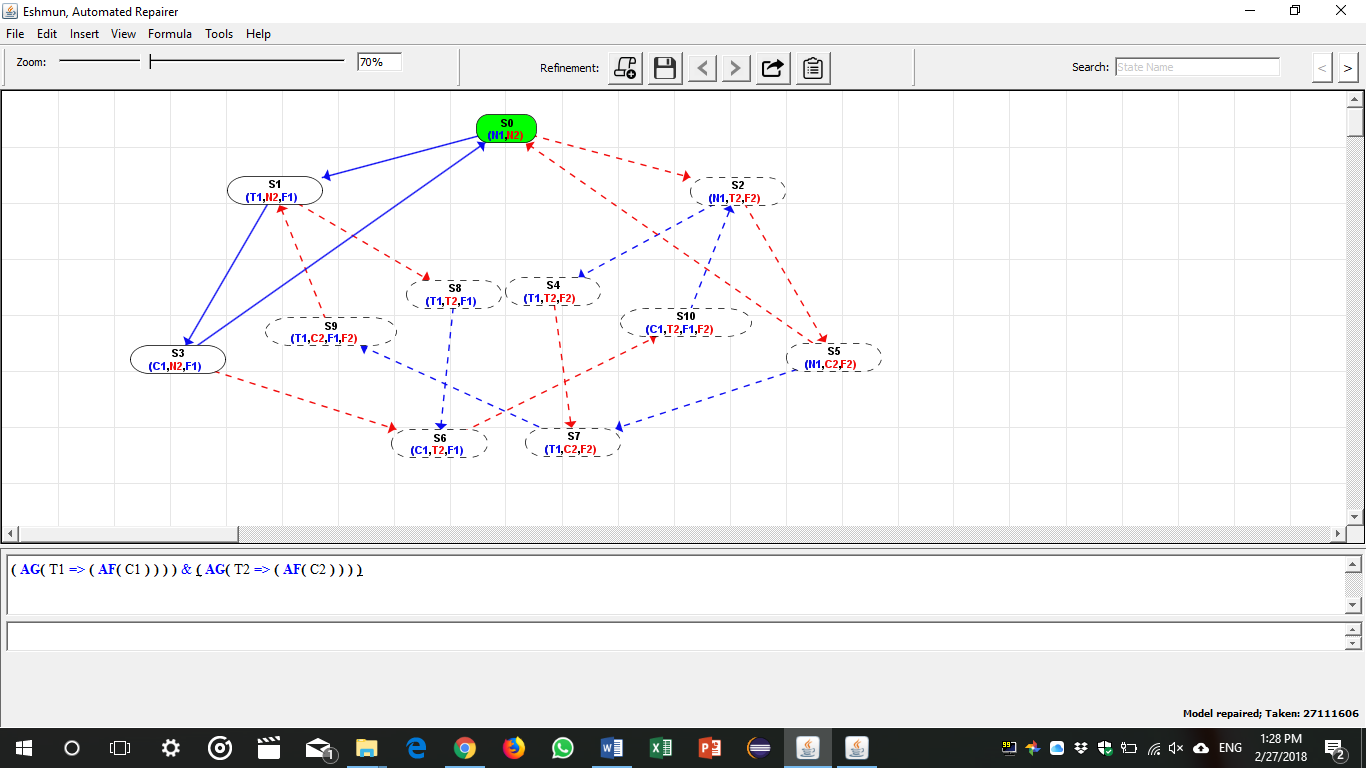
1. I started adding F1 and F2 respectively to the states I already have. I obtained the following:

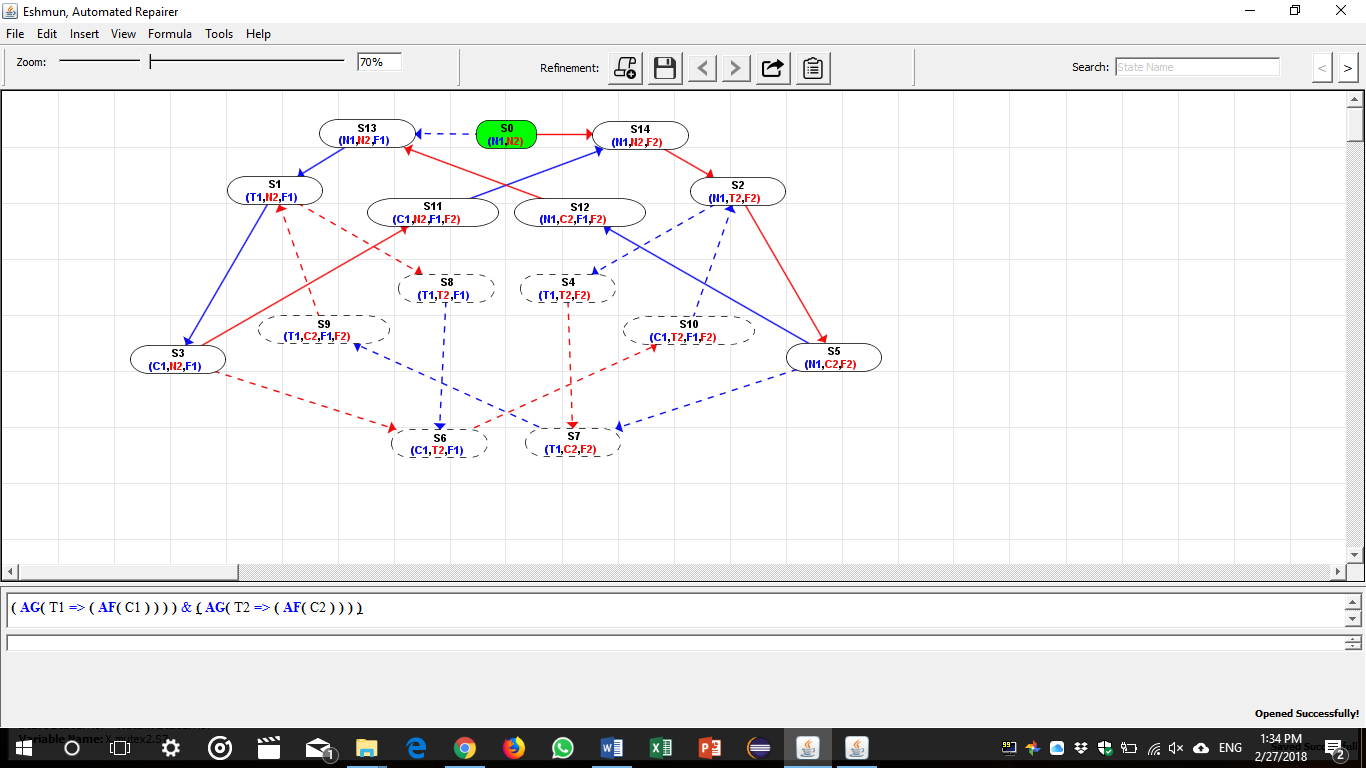


1. At this point it is clear that state S4 should be split into 2 states. (T1, T2, F1) and (T1, T2, F2)

In addition, an intermediate state is needed to fix the transition between state S7 to S1 and S6 to S2.



1. Now, after getting all the transitions right. I tried to model check for liveness property. I obtained the following:  
   
2. It is clear that the transitions from state S3 to S0 and from state S5 to S0 does not guarantee liveness since either processes will be able to execute the critical section infinitely without giving the other process the chance to execute the critical section. To fix that I did the following:



The Model now is Live and has no Black transitions.

**Problem 2:**

We have f = O1 O2 O3 O4 O5  ….. ON

Observations:

* G+ (set of succeeding G) can be reduced to G. (example: GGGp is reduced to Gp)
* F+ (set of Succeeding F) can be reduced to F, (example: FFFp is reduced to Fp)

This will implies that f = O1 O2 O3 O4 O5  ….. ON can be reduced to (GF)+ and optional G at the end or (FG)+ and an optional F at the end

This implies f is either one of the following:

* f = GF….GFp
* f = FG….FGp
* f = GF…GFGp
* f= FG…FGFp

other observations:

* FGFp can be reduced to FGp
* GFGp can be reduced to FGp

Therefore, f = O1 O2 O3 O4 O5  ….. ON  can be reduced to either:

1. f = FGp
2. f = GFp