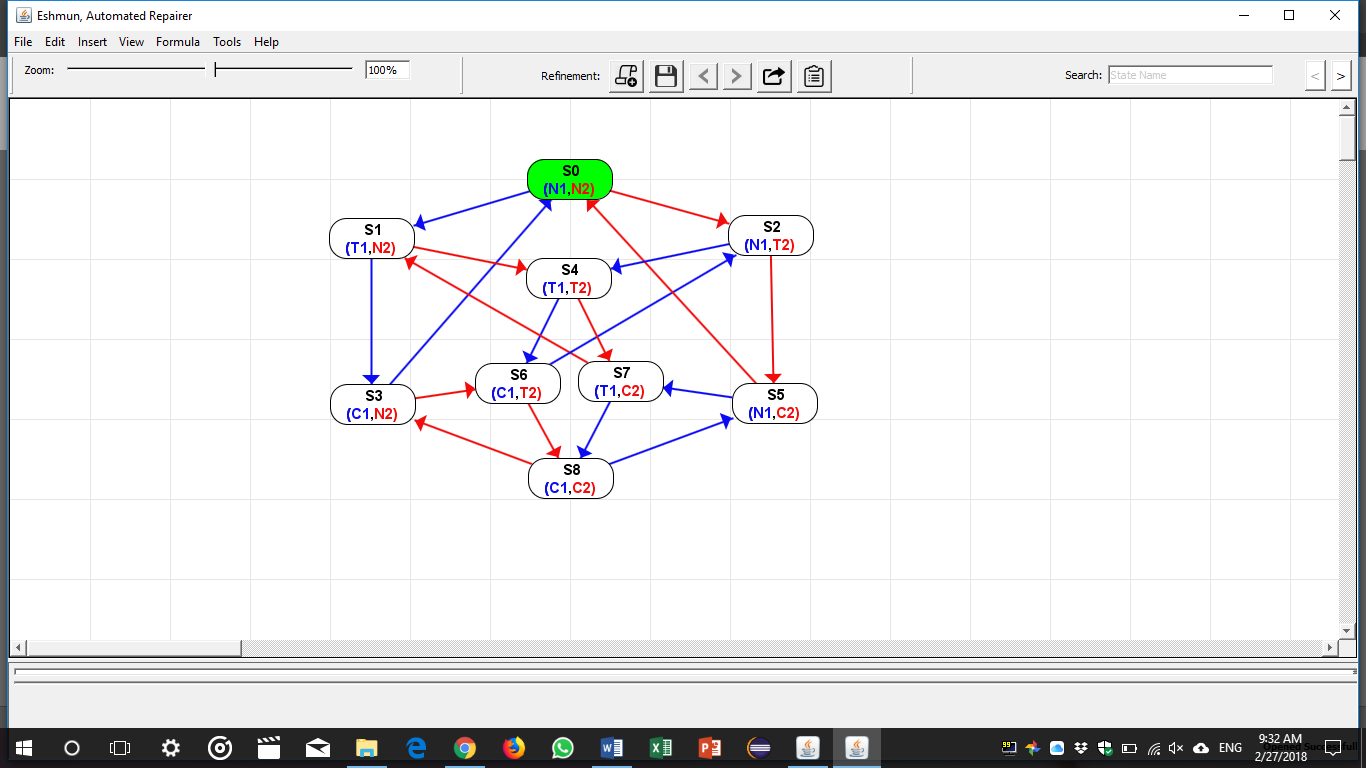
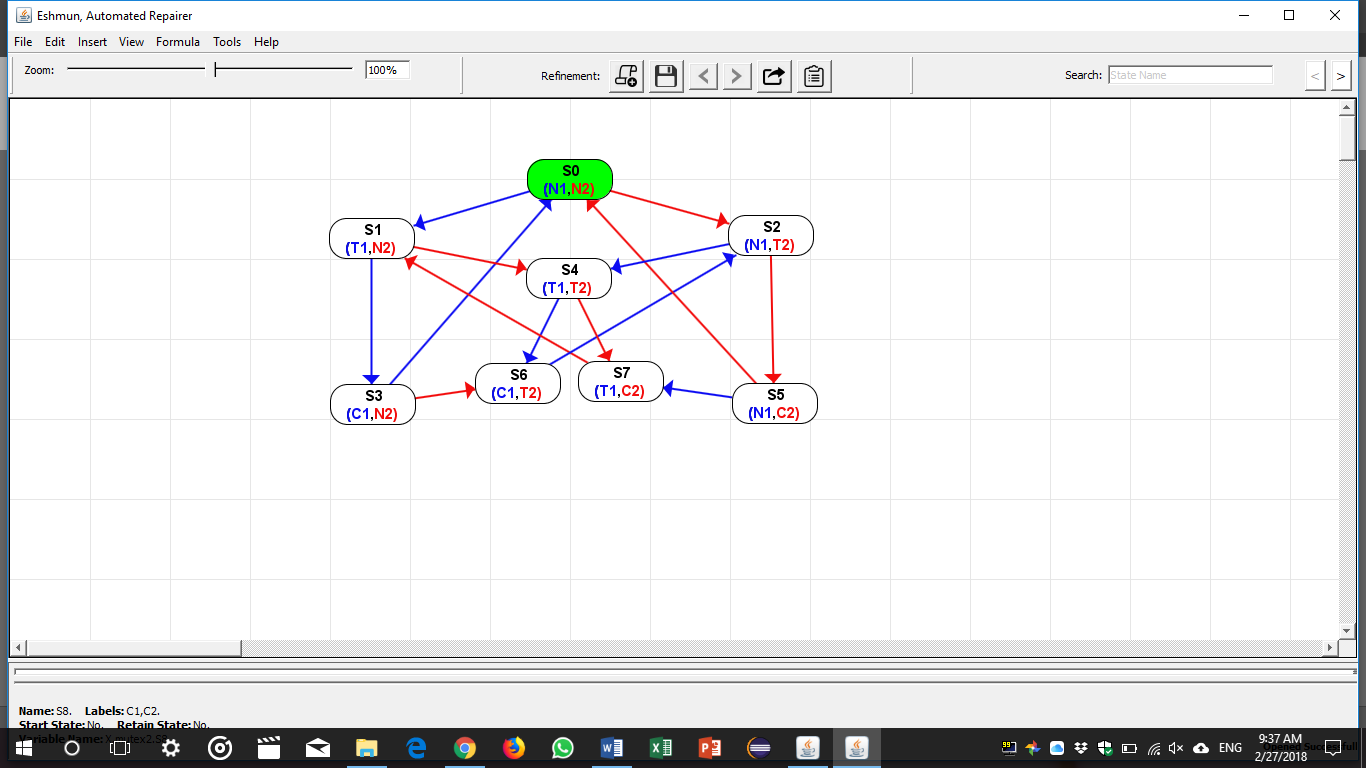
**Solving Mutex Liveness Problem (Procedure):**

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.

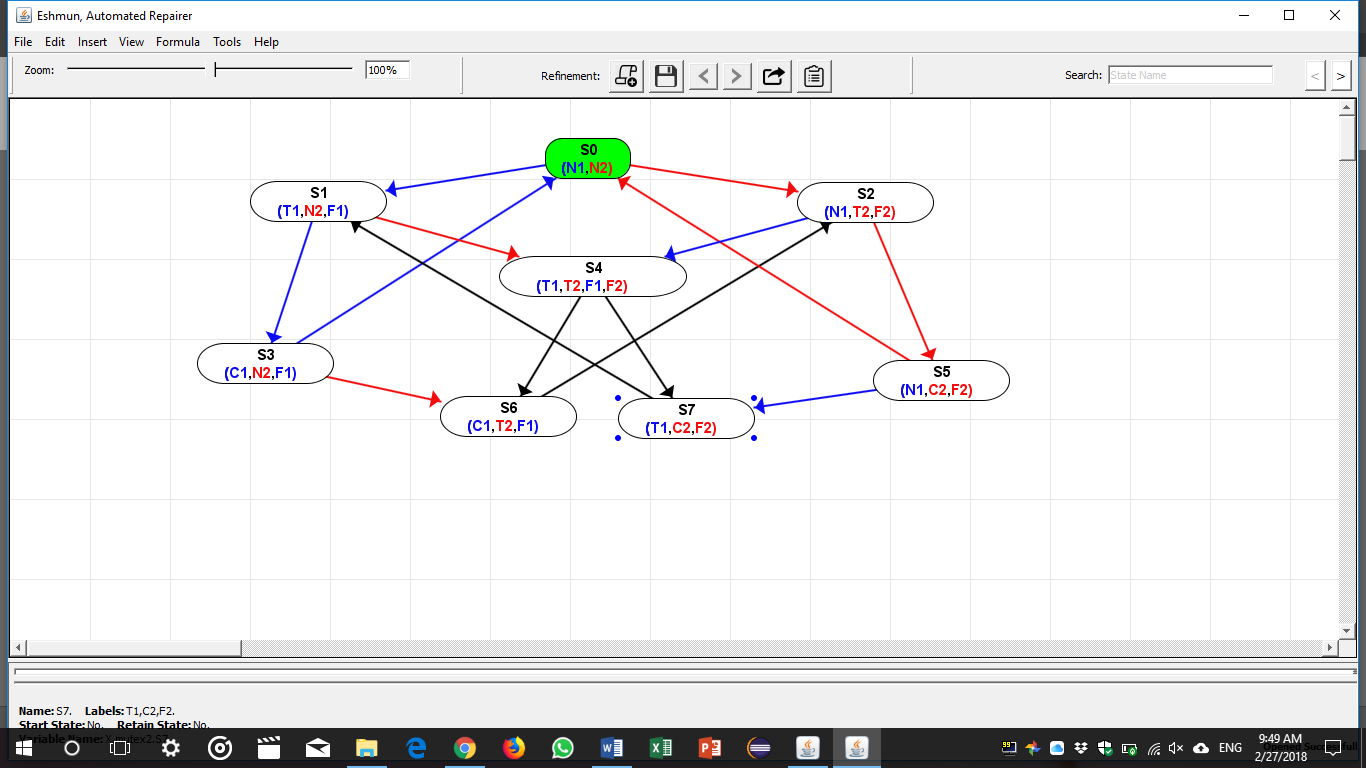
To do this, we will need a variable F (consider it as a lock) so that you cannot enter the critical section unless you have your lock.

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.

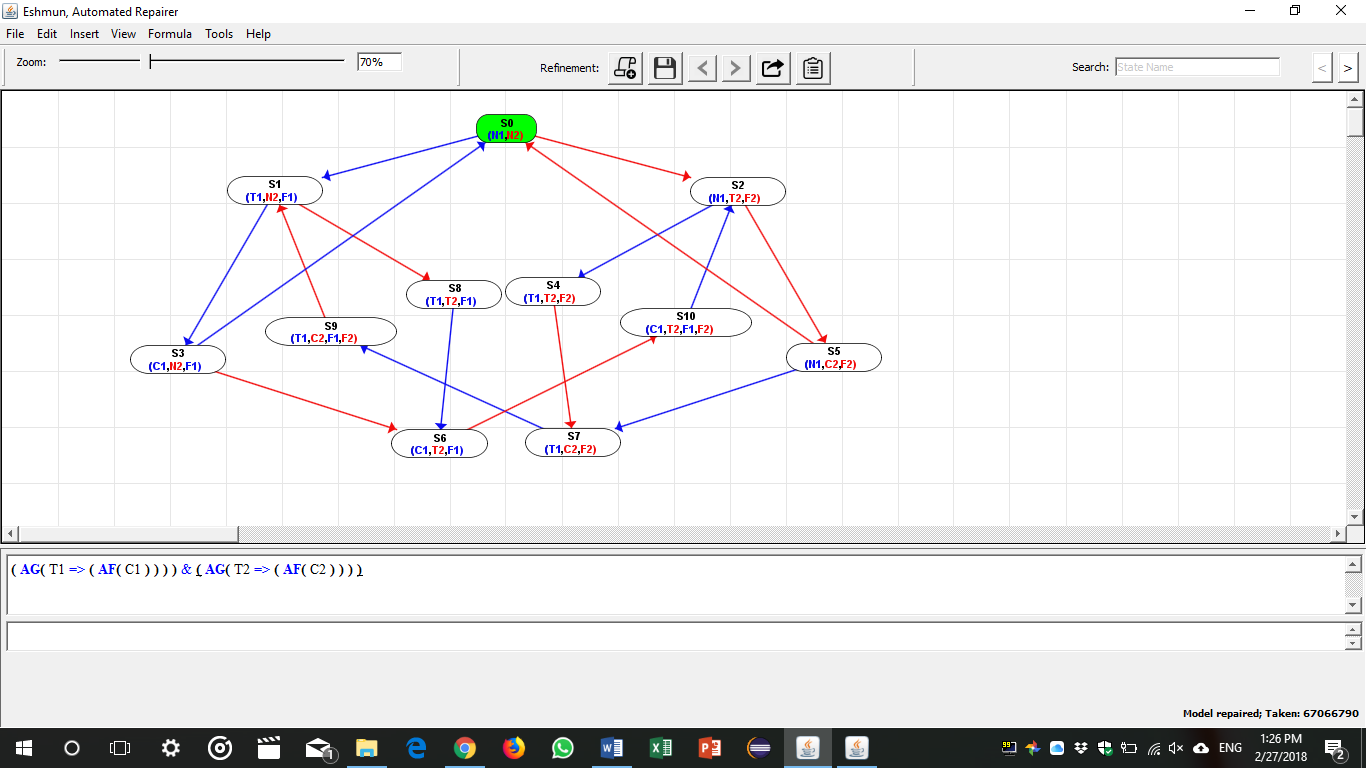
Observation: We might need up the 16 states (42) since every process now will have 4 different states (N, T, C, F)

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

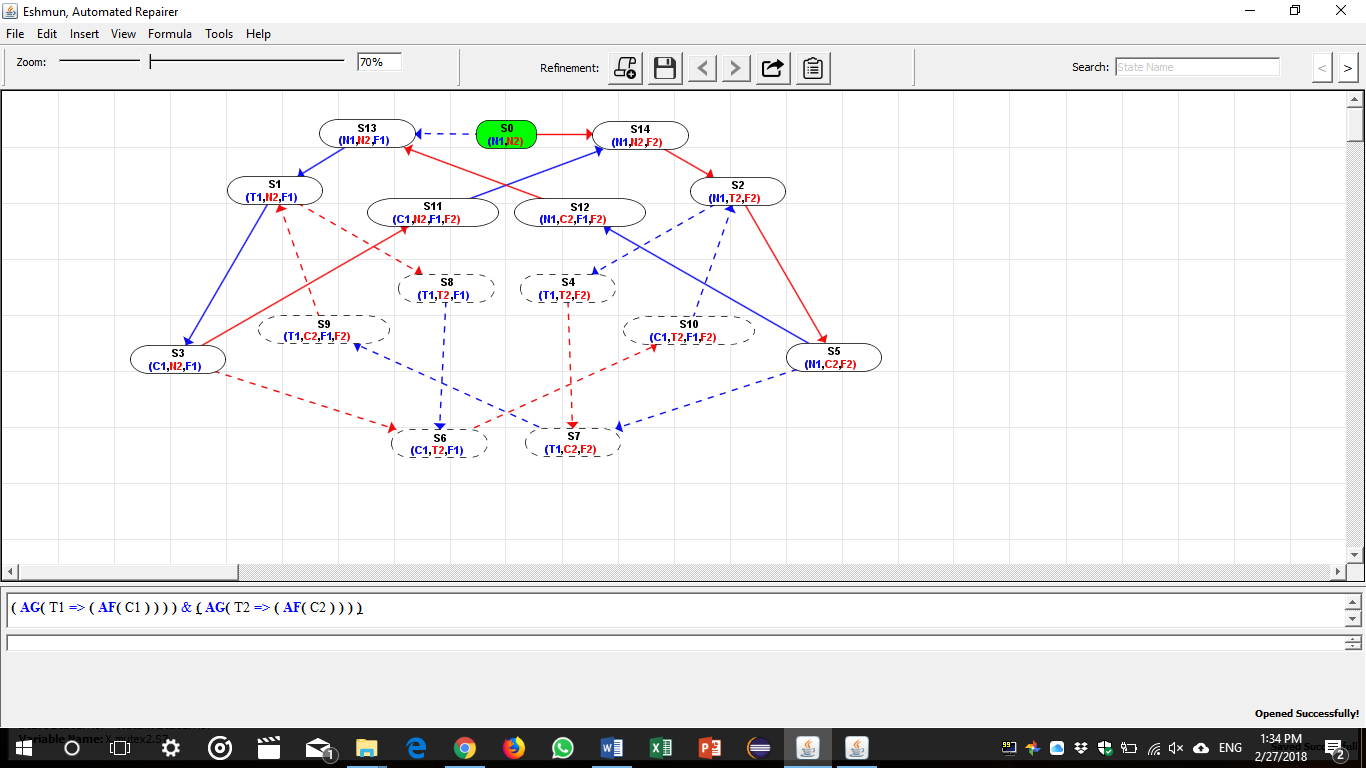


1. At this point it is clear that state S4 should be split into 2 states. (T1, T2, F1) and (T1, T2, F2)

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



1. 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:



1. The Model now is Live and has no Black transitions. But we still have a problem we are forcing each process to wait for the second process to execute the critical section before we allow it to enter its critical section again (The processes are forced to alternate).
2. We have reached this state because the CTL formula ( AG( T1 => ( AF( C1 ) ) ) ) & ( AG( T2 => ( AF( C2 ) ) ) ) that we are using is incomplete. It doesn’t state that we should allow every process to cycle if the other process is not doing anything (for example, Process p1 should be allowed to enter the CS infinitely as long as Process p2 is at state N2).
3. To fix this problem, we fix the CTL formula by adding ( EG( N1 ) ) & ( EG( N2 ) ) so that it will become as follows:

( AG( T1 => ( AF( C1 ) ) ) ) & ( AG( T2 => ( AF( C2 ) ) ) ) & ( EG( N1 ) ) & ( EG( N2 ) )

Now, it is clear and straight forward that adding 2 transitions (1 transition from State S3 to S0 and another transition from State S5 to S0 will solve the problem and make the model true.

