Deadline

The deadline of this homework is Apr. 24th at 23:59. Please submit a .zip file electronically on BlackBoard with:

- The required files specified in the UPPAAL part
- Your answer sheet including the fault trees and the form with name "CS132_HW2_YourName.pdf".

You should write all of your answers in the Answer Sheet provided below.

Answer Sheet: Risk Management Part

Construct the 6 fault trees mentioned above, and fill in the probabilities in the form below.

Define several events:

N1H: AOA is normal while sensor 1 report AOA high.

N2N: AOA is normal and sensor 2 report AOA normal as expected.

N1N: AOA is normal and sensor 1 report AOA normal as expected.

N2H: AOA is normal while sensor 2 report AOA high.

H1L: AOA is high while sensor 1 report AOA low.

H2L: AOA is high while sensor 2 report AOA low.

H1H: AOA is high and sensor 1 report AOA high as expected.

H2H: AOA is high and sensor 2 report AOA high as expected.

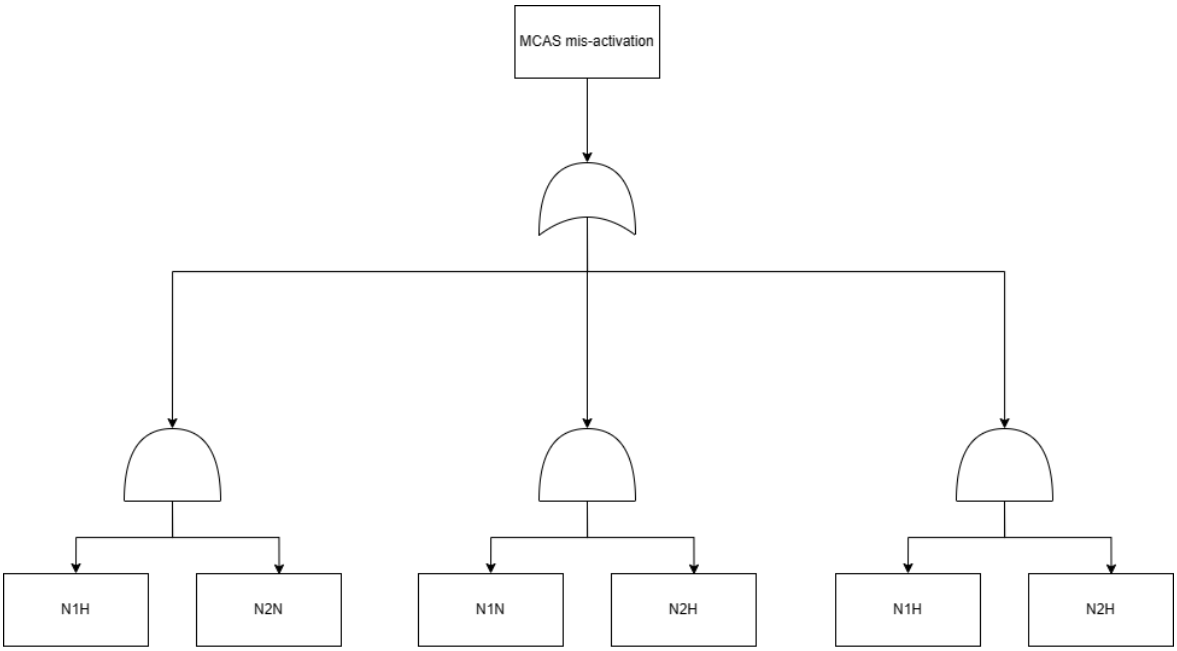
N3H: AOA is normal while sensor 3 report AOA high.

N3N: AOA is normal and sensor 3 report AOA normal as expected.

H3H: AOA is high and sensor 3 report AOA high as expected.

H3L: AOA is high while sensor 3 report AOA low.

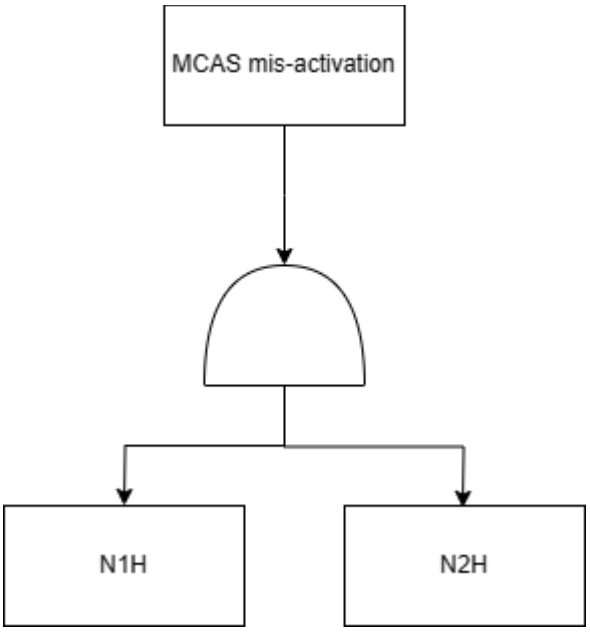
Fault Tree Analysis on the faulty design



Calculate probabilities:

0.01*0.99+0.01*0.99+0.01*0.01=1.99%

A More Aggressive Design



Calculate probabilities:

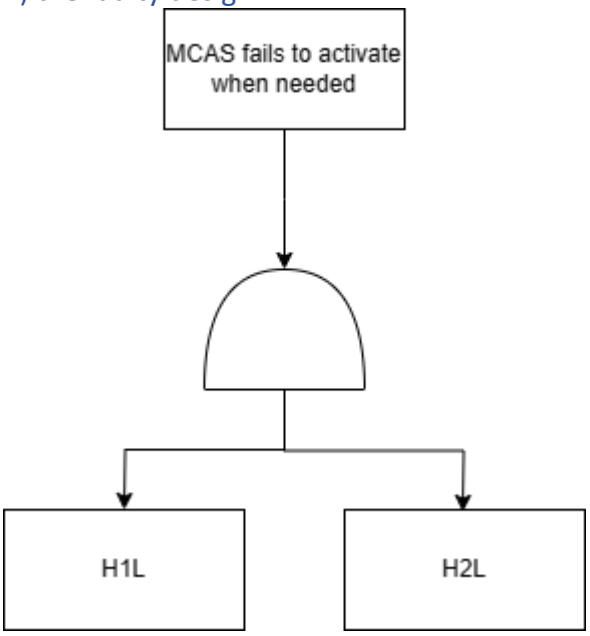
0.01*0.01=0.01%

Analyze:

Since 0.01%<1.99%, so it's clear that a more aggressive design does reduce the probability for MCAS mis-activation.

Balancing False-Positives and False-Negatives

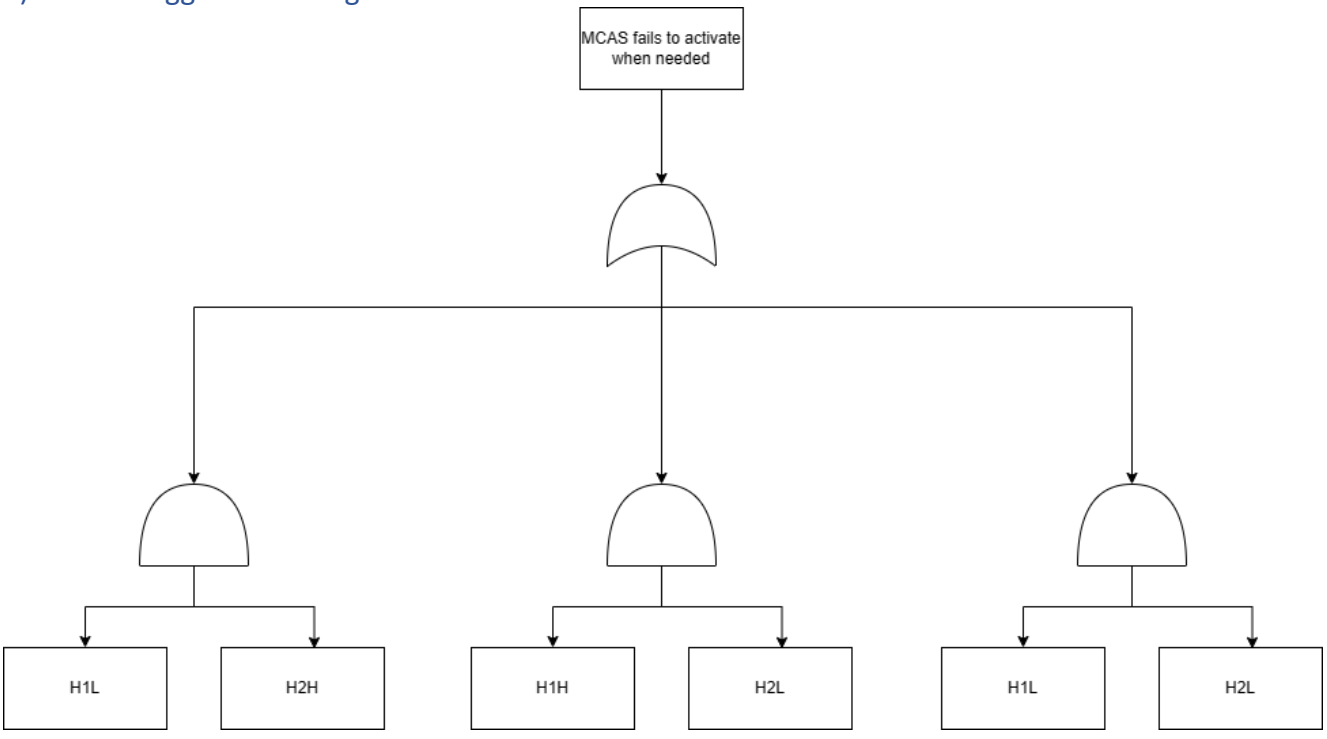
1) the faulty design



Calculate probabilities:

0.01*0.01=0.01%

2) the new aggressive design

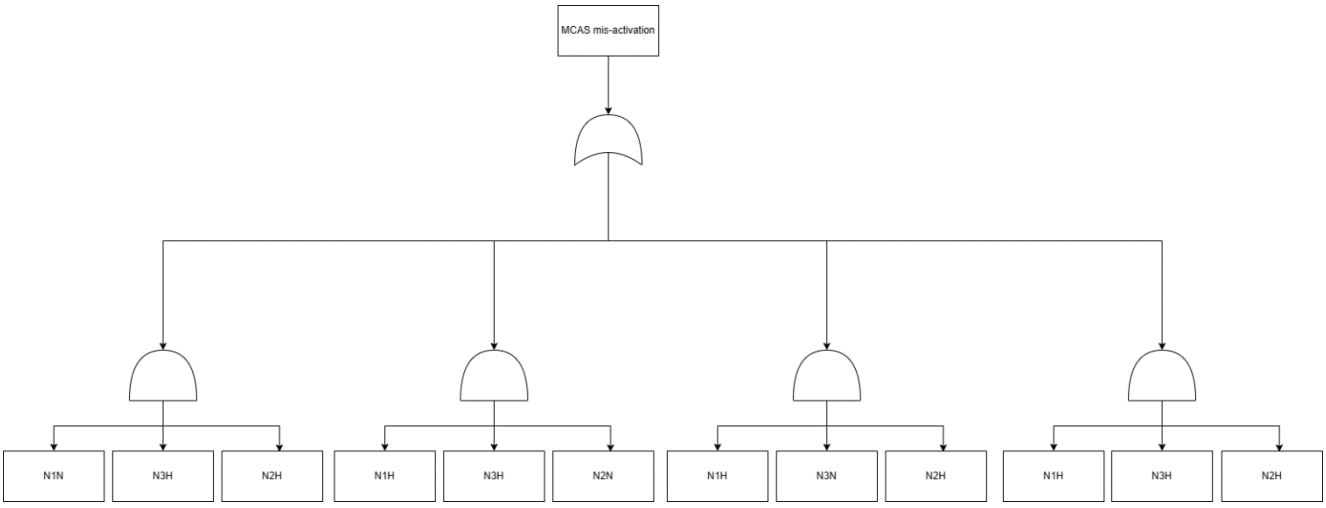


Calculate probabilities:

0.01*0.99+0.01*0.99+0.01*0.01=1.99%

Adding Another AOA Sensor

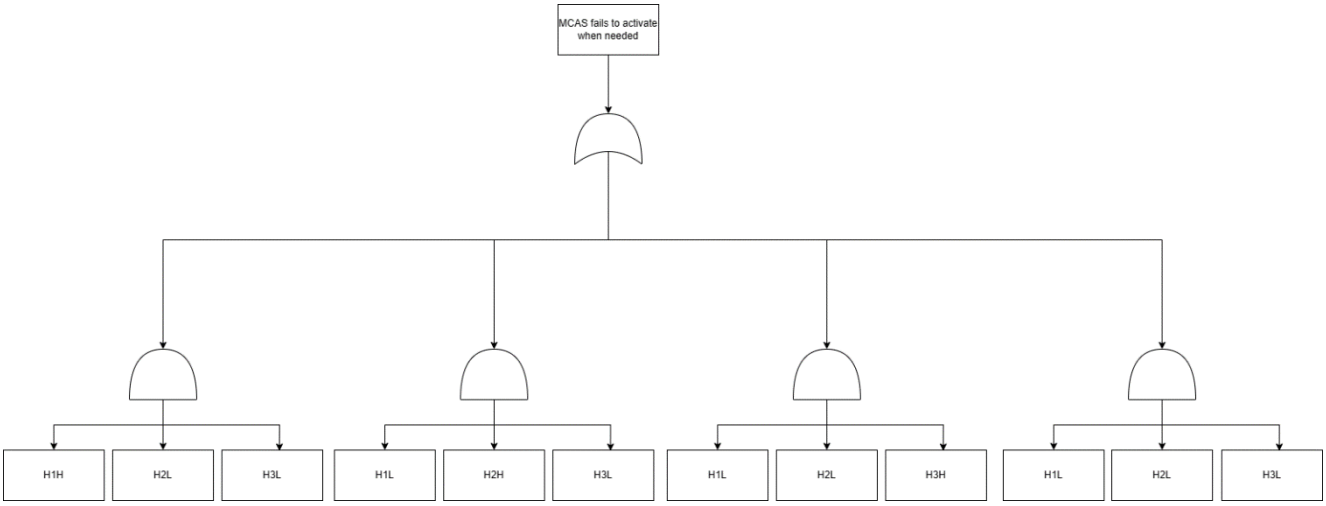
MCAS mis-activation when the actual AOA is normal



Calculate probabilities:

0.01*0.01*0.99*3+0.01*0.01*0.01=0.0298%

MCAS fails to activate when needed



Calculate probabilities:

0.01*0.01*0.99*3+0.01*0.01*0.01=0.0298%

Analyze:
As for MCAS mis-activation, adding another AOA sensor does reduce more possibilities for MCAS mis-activation compared to the faulty design, however, since 0.0298%>0.01%, so this method is slightly inferior compared to the aggressive design.

As for MCAS fails to activate when needed, adding another AOA sensor does reduce more possibilities for this situation compared to the aggressive design, however, since 0.0298%>0.01%, so this method is slightly inferior compared to the faulty design.

However, if we define 'accident rate' as the combination of MCAS mis-activation and MCAS fails to activate when needed, and we assume that the likelihood of both accidents is equal, then we can calculate the expected accident probability under the Adding Another AOA Sensor strategy, which is (0.0298% + 0.0298%) / 2 = 0.0298%. For the other two strategies, the expected accident probability is (1.99% + 0.01%) / 2 = 1.00%, both of which are higher than the Adding Another AOA Sensor strategy. Therefore, we can roughly consider the Adding Another AOA Sensor strategy to be a relatively good overall option.

	Faulty	Aggressive	2/3
MCAS mis-activation	1.99%	0.01%	0.0298%
MCAS fails to activate when needed	0.01%	1.99%	0.0298%

Answer Sheet: UPPAAL Part

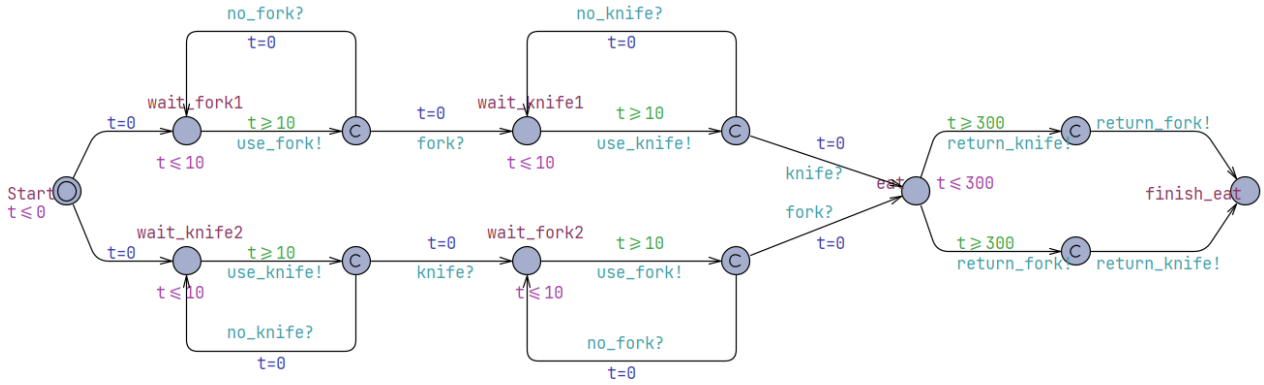
Required files: greedy.xml, greedy.xtr, polite.xml, and polite.xtr (if necessary)

1-a:
knife1_fork1: There is one knife and one fork on the table, both of them are available.
Knife0_fork1: There is only one fork on the table, and no knife on the table. Only the fork is available.
Knife1_fork0: There is only one knife on the table, and no fork on the table. Only the knife is available.
Knife0_fork0: There is no knife and no fork on the table, both of them are unavailable.

1-b:
knife0_fork1

1-c:
knife0_fork0

2-a: Model File: greedy.xml



2-b:

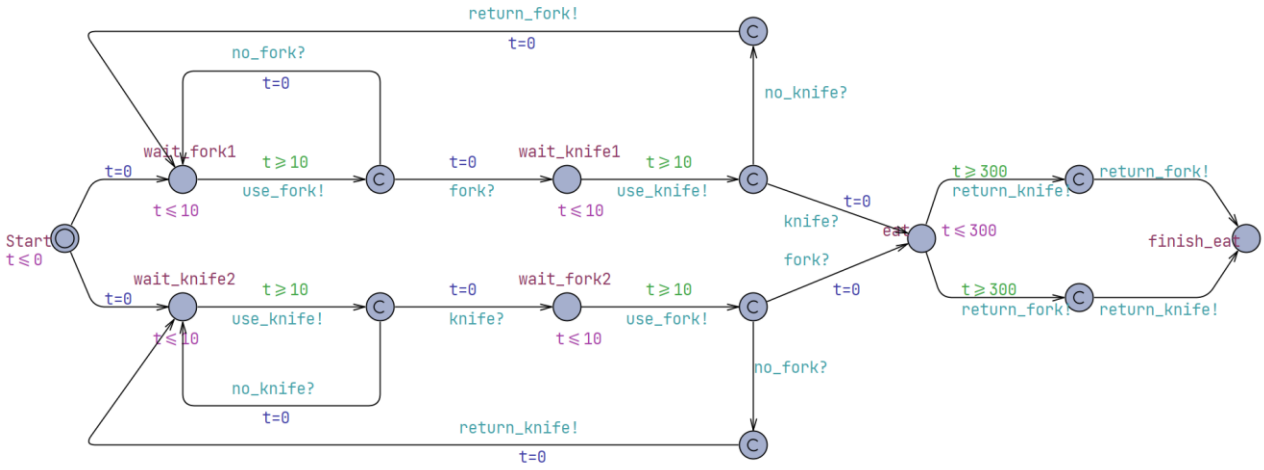
Query:
A<> (gclk > 700 && (GreedyCustomer1.finish_eat == true && GreedyCustomer2.finish_eat == true))

Validation Result:

```
A<> (gclk > 700 && (GreedyCustomer1.finish_eat == true && GreedyCustomer2.finish_eat == true))
验证费时/kernel费时/总费时: 0s / 0s / 0.006s.
常驻内存/虚拟内存的使用峰值: 10,940KB / 52,064KB.
不满足该性质.
```

Counter Example (Trace File): greedy.xtr
Please see this in the other part of the .zip file.

3-a: Model File: polite.xml



3-b:

Query:
E<> (gclk == 700 && (PoliteCustomer1.finish_eat == true && PoliteCustomer2.finish_eat == true))

Validation Result:

```
E<> (gclk == 700 && (PoliteCustomer1.finish_eat == true && PoliteCustomer2.finish_eat == true))
验证费时/kernel费时/总费时: 0s / 0s / 0.004s.
常驻内存/虚拟内存的使用峰值: 14,708KB / 60,088KB.
满足该性质.
```

Counter example (Trace File): polite.xtr (if necessary)
Both polite customers may finish their meal at 700s, so there is no counterexample.

CS132: Software Engineering
HW2: Risk Management

On Mar 10 2019, Ethiopian Airline ET302 from Addis Ababa to Nairobi crashed 6 minutes after takeoff, killing all 157 people on board. This accident and the previous Lion Air accident result in full investigation of the Boeing 737 Max model, and all aircraft with the same model are therefore grounded. In the lecture we analyzed how the tragedy happened. In this exercise, assume you work for Boeing, please do Fault Tree Analysis (FTA) on the incident, and analyze potential solutions to the problem. Note that there are 100 pt for this homework, but on blackboard the score will be divided by 10.

Fault Tree Analysis on the faulty design

We now know that the accidents happened because the planes were sent into nosedive by MCAS at low altitude, and the MCAS was mistakenly activated due to a faulty AOA sensor. Construct a fault tree and calculate the probability for "MCAS mis-activation when the actual AOA is normal".

Note: In this faulty design, there are two AOA sensors on each 737 Max aircraft, and MCAS is activated if one of the AOA sensors reports high AOA.

Note: For all the calculations in this homework, assume the probability for one AOA sensor to report high AOA when the actual AOA is not high is 1%, the probability for one AOA sensor to report low or normal AOA when the AOA is high is 1%, and the probability for an AOA sensor to report correct AOA is 98%.

A More Aggressive Design

Someone proposed that instead of activating MCAS after one of the sensors reports high AOA, requiring both sensors to report high AOA for MCAS activation may reduce the chance for MCAS mis-activation. Draw the fault tree for this new design and analyze whether the probability for MCAS mis-activation has been reduced.

Balancing False-Positives and False-Negatives

There is another risk that we must consider, that is "the MCAS fails to activate when needed". This may happen when AOA sensor reports low AOA when the AOA is high. Draw the fault tree for the hazardous situation "MCAS fails to activate when needed" for 1) the faulty design and 2) the new aggressive design and compare their probabilities.

Adding Another AOA Sensor

Someone also proposed to use 3 AOA sensors instead of 2 to increase redundancy. The MCAS now requires 2 AOA sensors to report high AOA in order to be activated. For this new design, draw FTA for both "MCAS mis-activation when the actual AOA is normal" and "MCAS fails to activate when needed", and discuss whether this is an overall better design.

Problem

Construct the 6 fault trees mentioned above, and fill in the probabilities in the form below.

	Faulty	Aggressive	2/3
MCAS mis-activation			
MCAS fails to activate when needed			

Please write your answer in Answer Sheet.

CS132: Software Engineering
HW2: UPPAAL – Dining Customers

Dining philosophers problem is a typical synchronization problem in computer science. In this homework, a simple version - Dining Customers will be analyzed. We will see how UPPAAL reveals the bad behavior of greedy customers, and how it demonstrates a good strategy for polite customers.

Dining Customers:

There are two customers sitting at a table, with a knife and a fork. They want to eat a meal. Customers will have a think for exactly 10s before he/she take a fork or a knife. A customer can eat the meal only when he/she has both the knife and the fork, and will finish the meal using exactly 300s. After finishing the meal, he/she will return the fork and the knife to the table immediately. Customers do not mind whether a tableware is used by others.

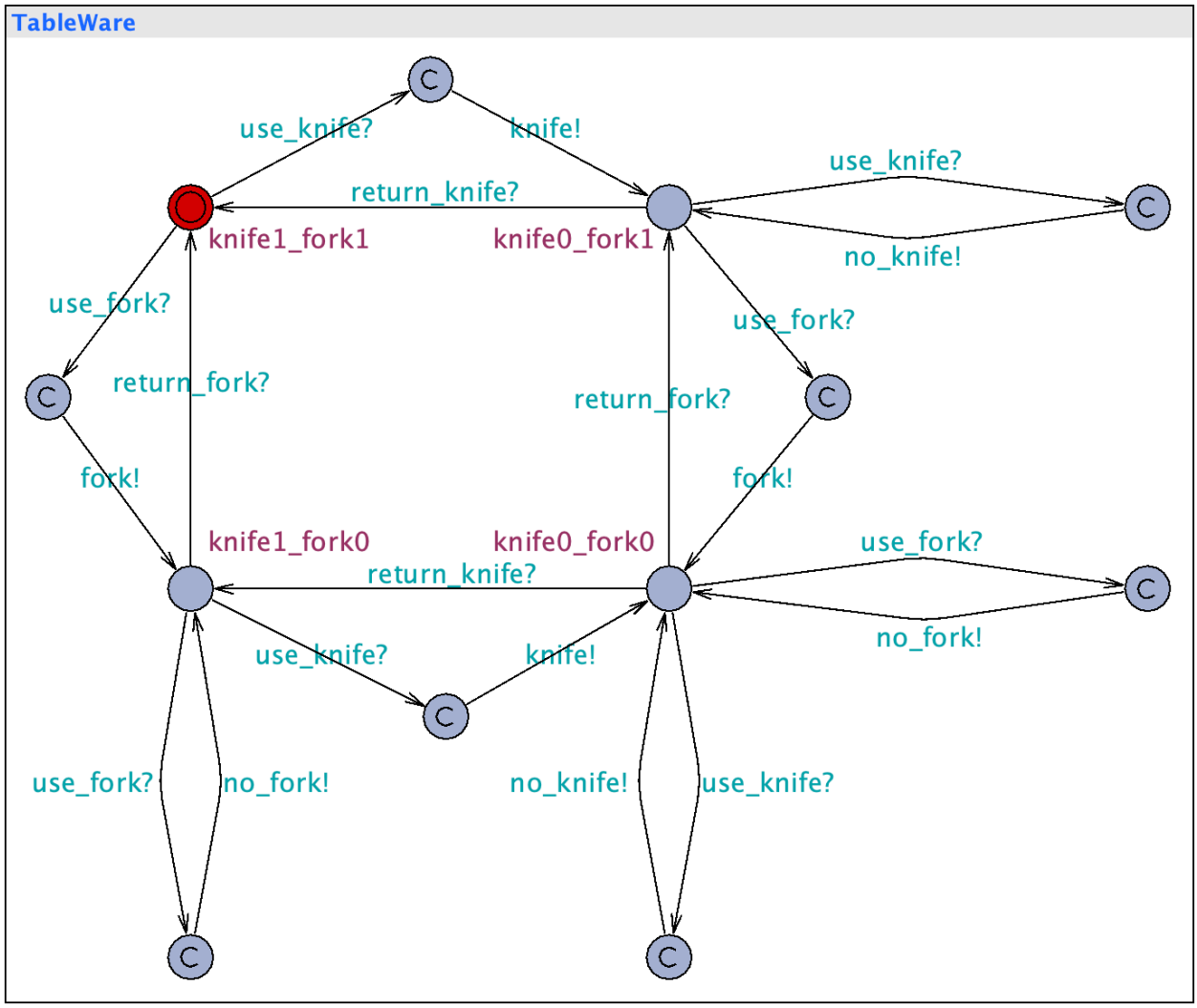
Greedy customers will ask for a tableware greedily. For example, if a greedy customer1 has a fork, he/she will ask for the knife immediately after thinking. If there is no knife, he/she will think for another 10s and ask for the knife again. If at the same time, greedy customer2 has a knife, he/she will behave similarly to get a fork. Therefore, both of them are waiting for the other tableware, but there is no tableware on the table.

For polite customers, if he/she asks for a tableware after thinking, and there is no such a tableware, he/she will release the resources in his/her hand, i.e. return the fork/knife to the table.

A template uppaal model is provided: `hw2_knife_fork.xml`.

Note that you can only use provided broadcast channels and provided templates. You should not modify the Declarations, TableWare template and the processes in System declarations.

1. Understand the model of tableware



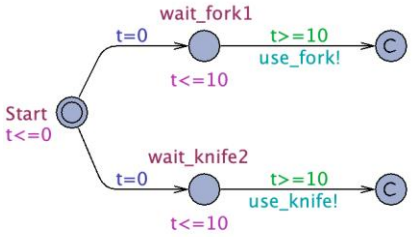
The model of tableware is provided. There are four named states: knife1_fork1, knife0_fork1, knife1_fork0 and knife0_fork0.

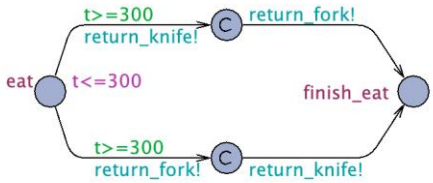
a. please explain the meaning of the four states.

b. if at the time 10s, both customer1 and customer2 asks for a knife, which state will the TableWare be in at the time 11s?

c. if at the time 10s, customer1 asks for a knife and customer2 asks for a fork, which state will the TableWare be in at the time 11s?

2. model greedy customer





a. The model of greedy customer misses a piece, try to complete this model.

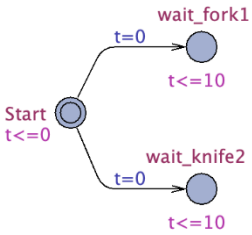
b. Find the counter example that neither of the two customers can finish the meal after 700s (e.g. reach the state finish_eat), because one of them occupies a fork while the other occupies a knife.

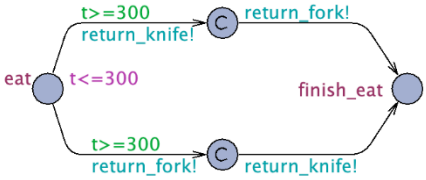
   Your query string:

   Validation Result (figure):

   Counter Example (Trace File): greedy.xtr

3. model polite customer





a. The model of polite customer misses a piece, try to design a polite customer with this template.
b. Design a query string to show that your design of polite customers can ensure that both of the two customers can finish their meal at the time 700s (e.g. reach the state finish_eat).

   Your query string:

   Validation Result (figure):

   Counter example (Trace File): polite.xtr (if necessary)