CS5011 Artificial Intelligence Practice: Assignment 4 Uncertainty

170004680

University of St. Andrews

May 2021

Contents

A Introduction

In this assessment, Bayesian Networks (BNs) have been modelled and used to reason with uncertainty. In part 1, a given problem scenario has been modelled and queried using BNs. In part 2, an algorithm to merge BNs has been implemented. In part 3, extensions to the basic specification have been added.

The solution for merging BNs can be executed according to the instructions provided in section A.2. The design and implementation of all parts is reported in section B.1. An evaluation of the BN solutions for each part is carried out in section B.2. Robust correctness testing is evidenced in section D.

A.1 Checklist

Table 1: Table indicating the completeness of this submission with respect to the specification requirements.

Part	**Section**	**Status**
1 - Bayesian Network Modelling	Report	Attempted, Fully Reported.
2 - Programming Bayesian Networks	Code	Attempted, Tested, Fully Working.
	Report	Attempted, Fully Reported.
3 - Extensions	Report	Attempted, Fully Reported.

A.2 Compilation & Execution Instructions

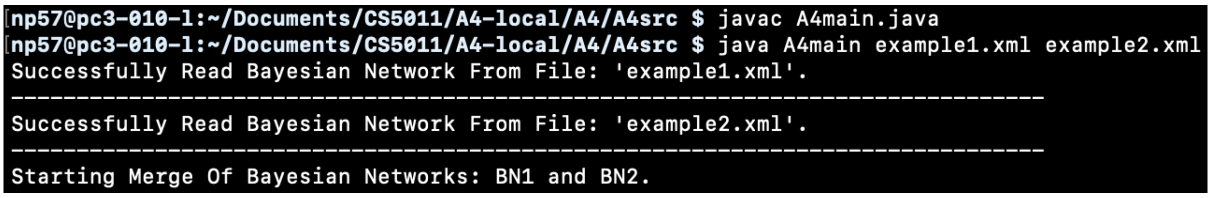
From within the `A4src/` directory, the `A4main` program, for merging Bayesian networks, can be compiled and executed with the following instructions (figure 1):

```
javac A4main.java
```

```
java A4main <BN1> <BN2>
```

, where <BN1> and <BN2> are relative paths from the `A4src/` directory to `XML` files representing Bayesian networks (as used by `bayes.jar`) to be merged.

Figure 1: The following screenshot shows the `A4main` program can be compiled and executed within the St. Andrews School of Computer Science lab environment without error.



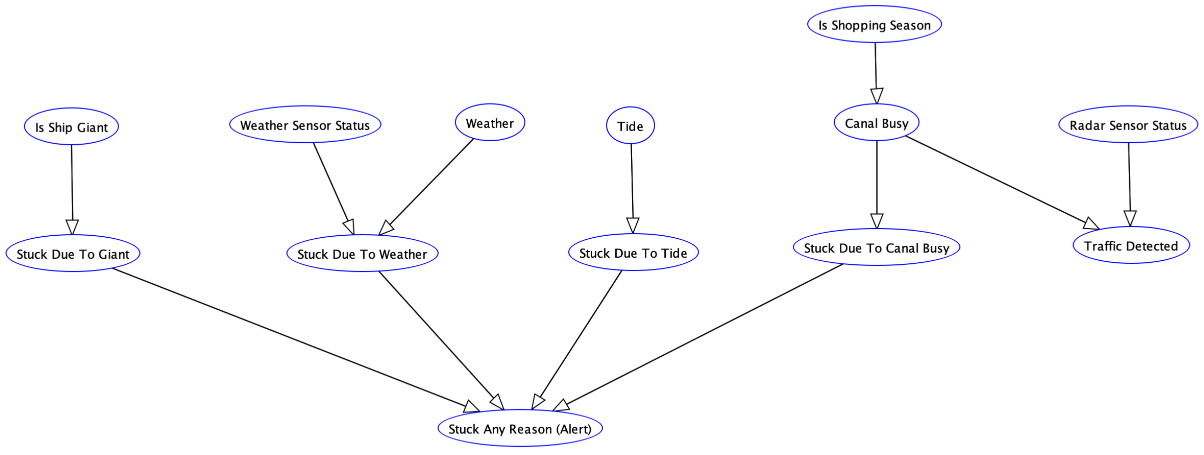
```
np57@pc3-010-l:~/Documents/CS5011/A4-local/A4/A4src $ javac A4main.java
np57@pc3-010-l:~/Documents/CS5011/A4-local/A4/A4src $ java A4main example1.xml example2.xml
Successfully Read Bayesian Network From File: 'example1.xml'.
----------------------------------------------------------------------------
Successfully Read Bayesian Network From File: 'example2.xml'.
----------------------------------------------------------------------------
Starting Merge Of Bayesian Networks: BN1 and BN2.
```

B Design, Implementation and Evaluation (Word Count: ∼ 1600)

B.1 Design and Implementation

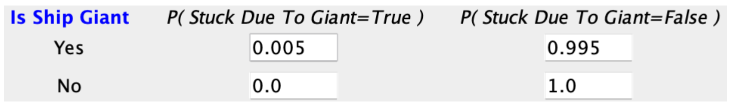
B.1.1 Part 1

Figure 2: Bayesian network developed for Part 1 using the specification. In all cases, the probabilities chosen for the nodes are likely lower in reality but have been set to allow for the network to trigger an alert with the given factors.



Most ships are giant ($P($`Is Ship Giant`$) = 0.75$). The canal is the main maritime route between the east and west, and long haul travel benefits from economies of scale. Given that a ship is giant, $P($`Stuck Due To Giant`$) = 0.005$ (figure 3). This is reasonable; hundreds of giant ships should make it through the canal without issue.

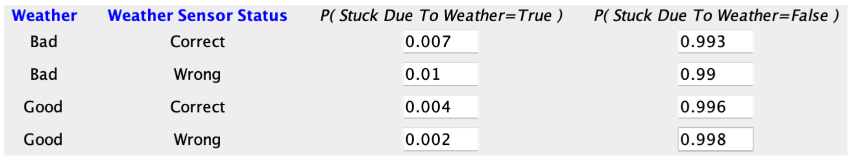
Figure 3: `Stuck Due To Giant` probability table. A non-giant ship certainly cannot be stuck due to being giant.



Is Ship Giant	P(Stuck Due To Giant=True)	P(Stuck Due To Giant=False)
Yes	0.005	0.995
No	0.0	1.0

The `Weather` can be either *Good* or *Bad*, occurring with equal chance (reasonable assumption without further domain knowledge of 'Mecyna'). The `Weather Sensor Status` is correct 99% of the time. A ship is less likely to get stuck in bad weather if the sensor is able to predict such weather as the ship can prepare. If the sensor predicts bad weather when the weather is good, then the ship is less likely to get stuck since it will be needlessly more alert (figure 4).

Figure 4: `Stuck Due To Weather` probability table.



Weather	Weather Sensor Status	P(Stuck Due To Weather=True)	P(Stuck Due To Weather=False)
Bad	Correct	0.007	0.993
Bad	Wrong	0.01	0.99
Good	Correct	0.004	0.996
Good	Wrong	0.002	0.998

The `Tide` can be *Low* or *High* ($P($`Tide`$=Low) = 0.25$). Low tide makes ships more likely to get stuck (figure 5), which is reflected by a ship being 10 times more likely to get stuck in low tide than high tide.

Figure 5: `Stuck Due To Tide` probability table.

Tide	P(Stuck Due To Tide=True)	P(Stuck Due To Tide=False)
Low	0.01	0.99
High	0.001	0.999

Canal busyness is affected by shopping seasons, occurring 20% of the time (figure 6). The canal is the main maritime route between the east and west, so the canal is likely busy most of the time, more so during shopping seasons. $P($`Stuck Due To Canal Busy`$|$`Canal Busy`$) = 0.01$ as ships should be used to travelling with traffic through the canal, but the chance of getting stuck is still non-zero.

Figure 6: `Canal Busy` (top) and `Stuck Due To Canal Busy` (bottom) probability tables. If the canal is not busy, a ship certainly can't be stuck for this reason.

Is Shopping Season	P(Canal Busy=Yes)	P(Canal Busy=No)
Yes	0.9	0.1
No	0.7	0.3

Canal Busy	P(Stuck Due To Canal Busy=True)	P(Stuck Due To Canal Busy=False)
Yes	0.01	0.99
No	0.0	1.0

The alert is represented by the '`Stuck Any Reason (Alert)`' node. A ship is certainly stuck for some reason if it is stuck due to any of the given factors. The probability the ship is stuck for some other reason is 0.001. If $P($`Stuck Any Reason (Alert)`$) > 0.03$, then this indicates an alert.

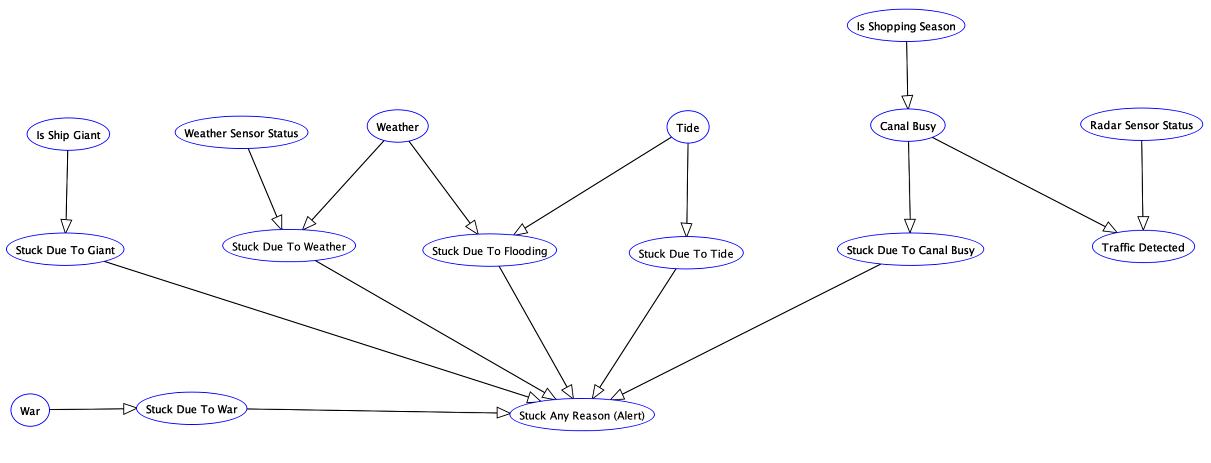
`Traffic Detected` (figure 7) is dependant on canal busyness and on the correctness of `Radar Sensor Status` (i.e., the sensor counting ships). The radar sensor is correct 90% of the time. If the radar sensor is correct, then detecting high traffic is certain when the canal is busy and impossible otherwise. If the radar sensor is incorrect (i.e., faulty at counting), then whether high traffic is detected or not is equally likely. It cannot be determined whether the sensor counts too many of too few ships and to what extent.

Figure 7: `Traffic Detected` probability table.

Canal Busy	Radar Sensor Status	P(Traffic Detected=High)	P(Traffic Detected=Low)
Yes	Correct	1.0	0.0
Yes	Wrong	0.5	0.5
No	Correct	0.0	1.0
No	Wrong	0.5	0.5

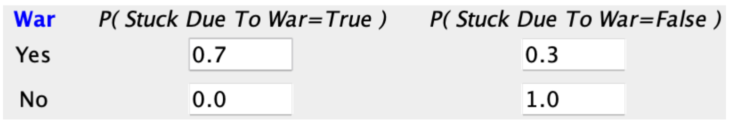
An extension with two additional events, flooding and war, has been created: `CANAL2` (figure 8).

Figure 8: Bayesian network, `CANAL2`, which extends the original BN created from the specification.



The event of `War` breaking out in 'Mecyna' has been given a probability of 0.001, which could be higher given the strategic value of the region. $P(\texttt{Stuck Due To War}|\texttt{War}) = 0.7$ as ships are likely to get stuck due to danger, obstruction for strategic advantage, etc. However, a ship is not guaranteed to get stuck if war breaks out as ships going to allied or neutral territories may be allowed passage.

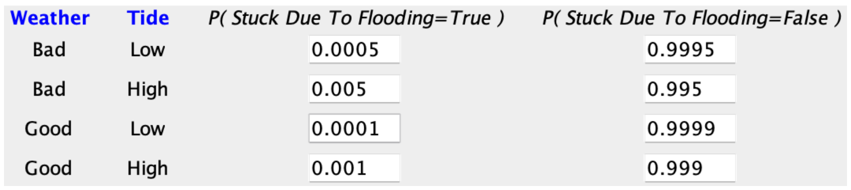
Figure 9: `Stuck Due To War` probability table.



War	P(Stuck Due To War=True)	P(Stuck Due To War=False)
Yes	0.7	0.3
No	0.0	1.0

Canal flooding is represented by '`Stuck Due To Flooding`' (figure 10). Ships are more likely to get stuck when there is flooding (e.g., passage denial for safety reasons). Flooding is more likely when there is bad weather and high tide. Tidal level is considered a larger factor than weather; the heaviest rain is unlikely to make up the difference in water level between low and high tide at the canal.

Figure 10: `Stuck Due To Flooding` probability table.



Weather	Tide	P(Stuck Due To Flooding=True)	P(Stuck Due To Flooding=False)
Bad	Low	0.0005	0.9995
Bad	High	0.005	0.995
Good	Low	0.0001	0.9999
Good	High	0.001	0.999

B.1.2 Part 2

The `A4main` program effectively has three stages: parsing the input XML files (representing the BNs to be merged) into memory, applying the implemented merging algorithm given by the lecture material, and writing the resulting merged BN to an output `XML` file. The output file name has the format, '`<BN1_name>_<BN2_name>_Merge.xml`', and is written to `A4src/`.

A Bayesian network is modelled as a mapping of variable names to custom variable objects, allowing for easy retrieval of a variable instance in either of the two networks being merged. This simplified the implementation of the merging algorithm. The identification of the non-intersection, internal, and external variables simply uses set operations on the variable names and retrieves any required attributes of the variables (e.g., dependencies) via the mapping.

The custom variable data type models variables similarly to the `XML` files used by `bayes.jar`, allowing for simple interfacing between the AISpace tool and the merging algorithm. Variables have a name, a list of outcomes (assumed binary in this assessment), and a position (i.e., location on the AISpace tool canvas). A variable also has a list of parent variable names, which models the dependencies in a BN, and a conditional probability table, given by a list of probabilities (same as the AISpace tool). The dimensions of the probability table are implied by the variable's parents and their outcomes.

B.2 Evaluation

B.2.1 Part 1

Diagnostic Query - Observing that '`Stuck Any Reason (Alert)`' is *True*, how likely are the weather and tide at cause for `CANAL1` and `CANAL2`?

- The probability the ship is `Stuck Due To Weather` increases from ~ 0.0055 to ~ 0.25 for `CANAL1` (figure 11) and to ~ 0.22 for `CANAL2` (figure 12). The probability the ship is `Stuck Due To Tide` increases from ~ 0.0033 to ~ 0.15 for `CANAL1` and to ~ 0.13 for `CANAL2`.
  The increase in probabilities of the ship being stuck due to weather or tide is lesser for `CANAL2` because `CANAL2` models additional factors, flooding and war, which share in the probability distribution of the reasons for the ship being stuck.

- The probability the `Weather` is bad increases from 0.5 to ~ 0.53 for `CANAL1` (figure 11) and to ~ 0.56 for `CANAL2` (figure 12). The probability increases are greater for `CANAL2` because the additional flooding factor in `CANAL2` is affected by the weather; flooding is more likely during bad weather.
  The probability of low `Tide` increases from 0.25 to ~ 0.33 for `CANAL1` and to ~ 0.30 for `CANAL2`. The probability increase is lesser for `CANAL2` because flooding, as modelled by `CANAL2`, is more likely to occur during high tide. This reduces the probability of low tide compared to `CANAL1`.

Figure 11: `CANAL1` before the diagnostic query (left) and after (right).

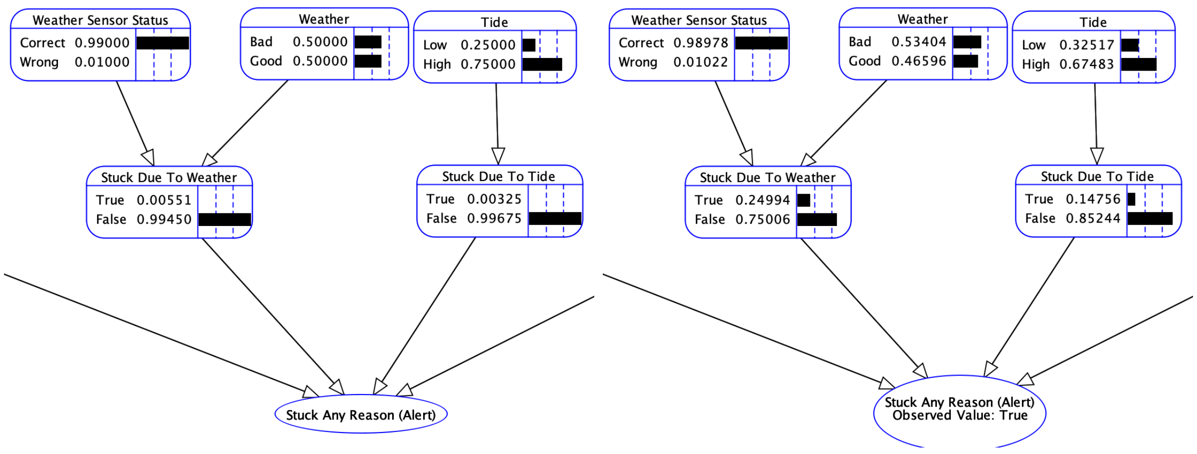
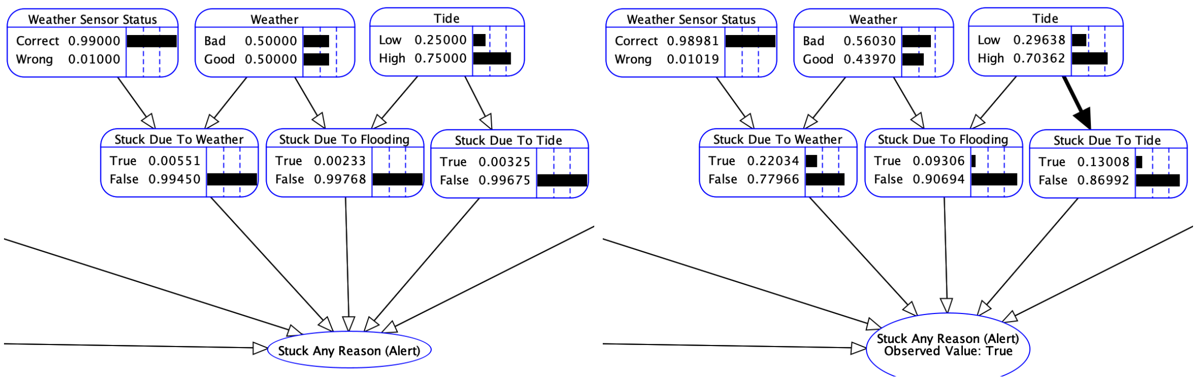


Figure 12: `CANAL2` before the diagnostic query (left) and after (right).



Predictive Query - Observing that `Weather` is *Bad*, how likely is it that a ship is stuck for any reason for `CANAL1` and `CANAL2`?

- The probability that a ship is `Stuck Due To Weather` increases from ~ 0.0055 to ~ 0.0070 for both `CANAL1` and `CANAL2`. This is expected; the conditional probabilities for `Stuck Due To Weather` are identical in both networks.
- The probability that a ship is stuck for any reason increased from ~ 0.022 to ~ 0.024 for `CANAL1` and from ~ 0.025 to ~ 0.028 for `CANAL2`.
  The probability increase of a ship getting stuck for any reason is greater for `CANAL2` ($\sim 12\%$ increase) than for `CANAL1` ($\sim 9\%$ increase). In `CANAL2`, bad weather affects an additional cause of the ship being stuck: flooding. Thus, `Weather` has a greater affect on the probability distribution amongst the reasons a ship may get stuck in `CANAL2` than `CANAL1`, so its observation has a larger affect on the alert node in `CANAL2`.
  The probability (before observations) of a ship being stuck for any reason is higher for `CANAL2` (~ 0.025) than `CANAL1` (~ 0.022). `CANAL2` models additional factors which happen to make a ship getting stuck more probable in this case.

Figure 13: `CANAL1` before the predictive query (left) and after (right).

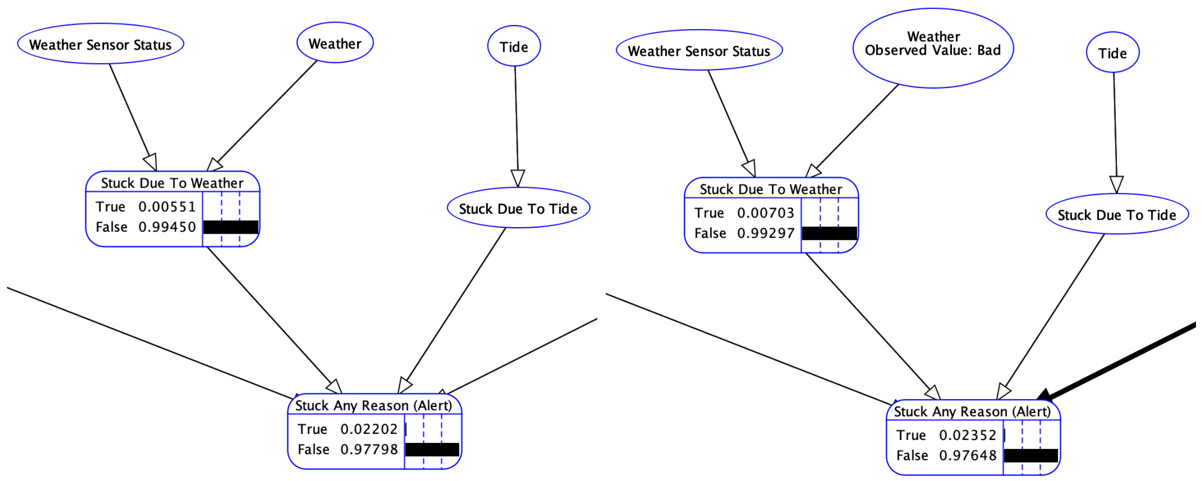
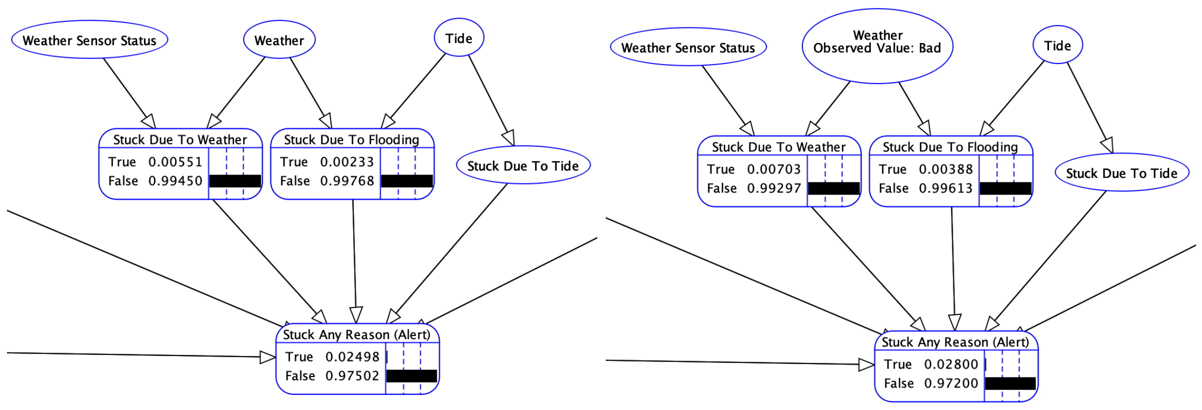


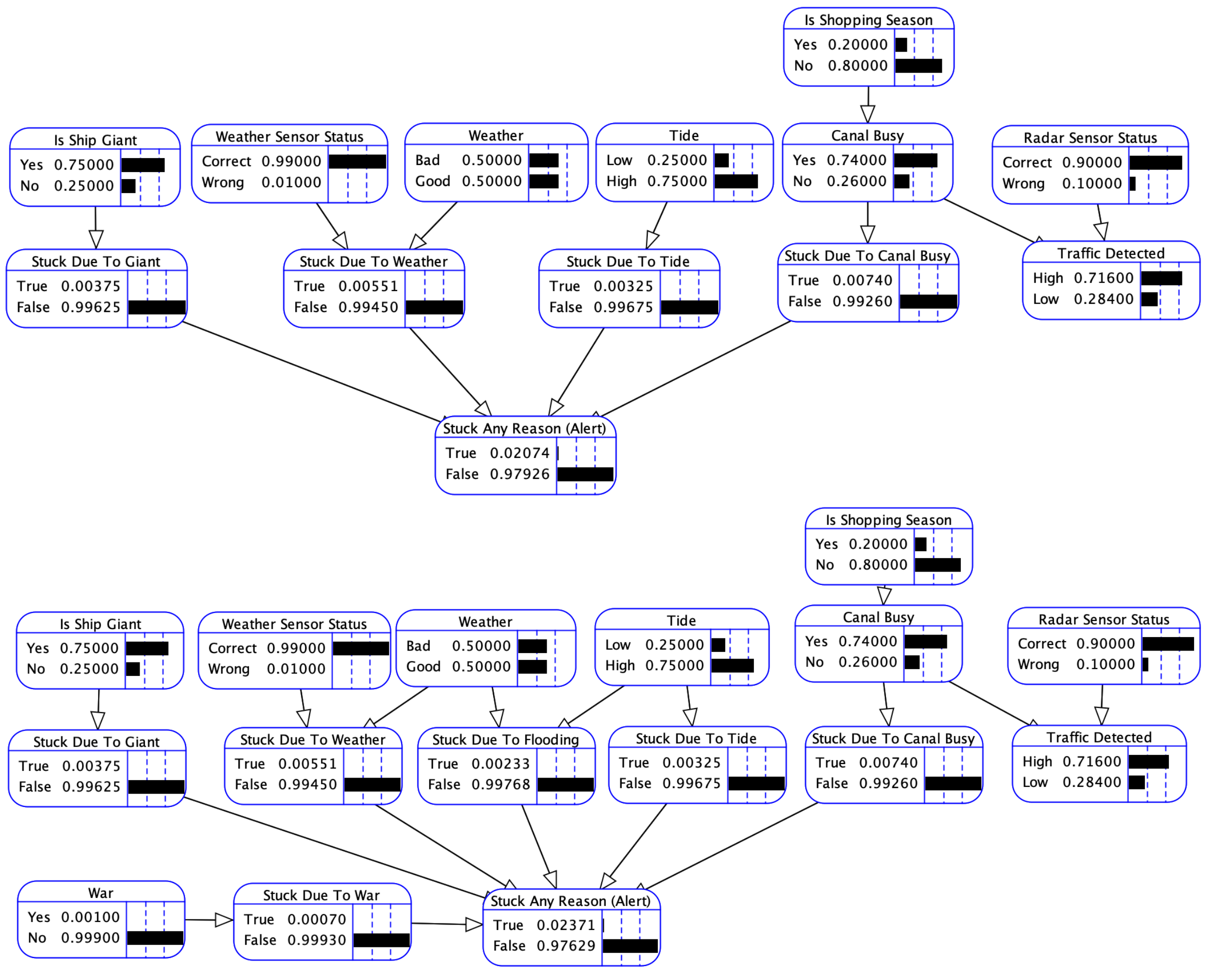
Figure 14: `CANAL2` before the predictive query (left) and after (right).



Profiling - Having made no observations, what are the likely conditions a ship will face? If a ship is stuck, what are the likely conditions the ship encountered? What are the most probable causes?

For both BNs, the probable scenario a ship experiences is (figure 15): the ship is giant-sized, experiences high tide and high traffic, and the weather is good or bad with equal likelihood. `CANAL2` also informs that the ship is likely experiences no war or flooding.

Figure 15: `CANAL1` (top) and `CANAL2` (bottom) without observations.



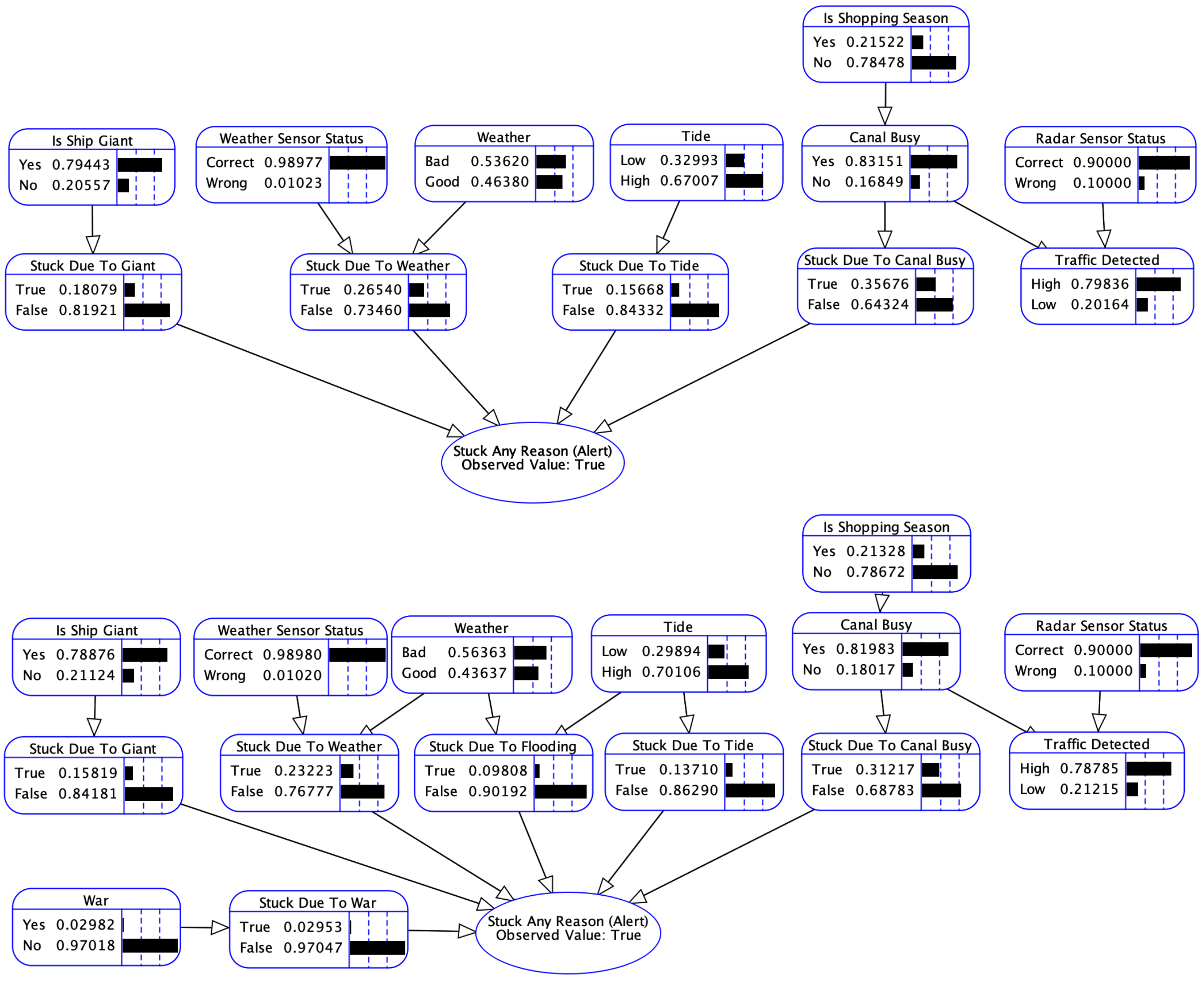
Given that a ship is stuck, both `CANAL1` and `CANAL2` show the ship was most likely giant and experienced bad weather, high tide, and high traffic. `CANAL2` also shows that the ship probably did not encounter war or flooding. Notably, the likelihood of the ship being giant-sized is less certain for `CANAL2` than for `CANAL1`, whilst bad weather and high tide is more certain, though the differences are small. This is true in general; the extension of `CANAL1` to `CANAL2` produces noticeable but minor differences, which is expected as the additional factors in `CANAL2` are extreme events with a significantly low chance of occurring.

For a stuck ship, the causes for `CANAL1` from most likely (figure 16) are: canal busyness, weather, ship is giant-sized, and low tide. This is similarly the case for `CANAL2`, where the new events, flooding and war, are the next most likely causes, respectively. Thus, the additional factors in `CANAL2` only minorly affect the likelihood of a ship getting stuck.

Further investigation, by observing that a ship is not stuck, reveals that the same conditions as if the ship were stuck are probable, except that the ship likely experienced good weather instead. This supports two analyses: a ship is likely to encounter conditions that could get it stuck (with these models), and the weather appears to be a distinguishing factor, which is expected.

A criticism of the models is that ship size should affect other factors, such as getting stuck due to the weather or tide. Giant ships are affected more by heavy winds (bad weather) and have a larger draft (more susceptible to low tide).

Figure 16: `CANAL1` (top) and `CANAL2` (bottom) with the observation: `Ship Stuck Any Reason (Alert)` = *True*.



B.2.2 Part 2 - Examples

The following figures show the final merged networks produced by the program for the three pairs of BNs given with this assessment: `example1.xml` and `example2.xml`, `CANAL1.xml` and `CANAL2.xml`, and `MEDICAL1.xml` and `MEDICAL2.xml`. For the third pair of BNs, `MEDICAL1.xml` was sourced from the lecture material and `MEDICAL2.xml` is a simple augmentation of `MEDICAL1.xml`. The program output associated with these results has also been provided in the appendix (section F), as requested.

Figure 17: The top and middle screenshots show the `example1.xml` and `example2.xml` Bayesian networks respectively. The bottom screenshot shows the program result of merging these two networks. The result is as expected and was verified further for correctness (i.e., by checking the probability tables).

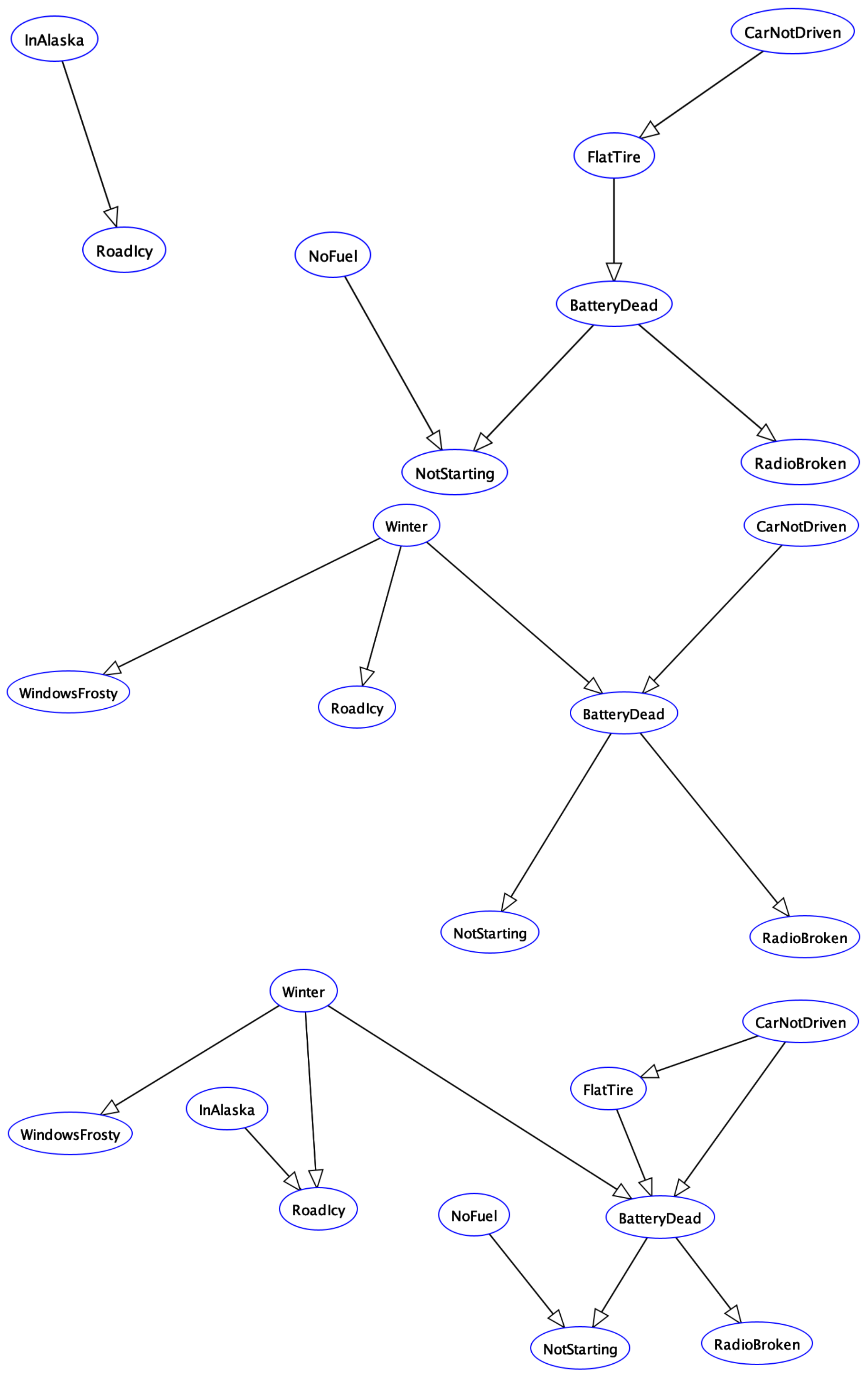
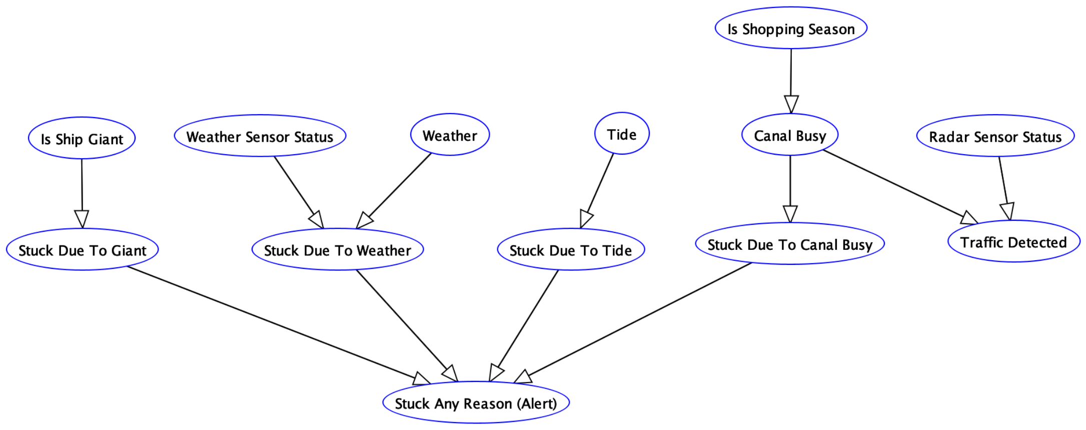
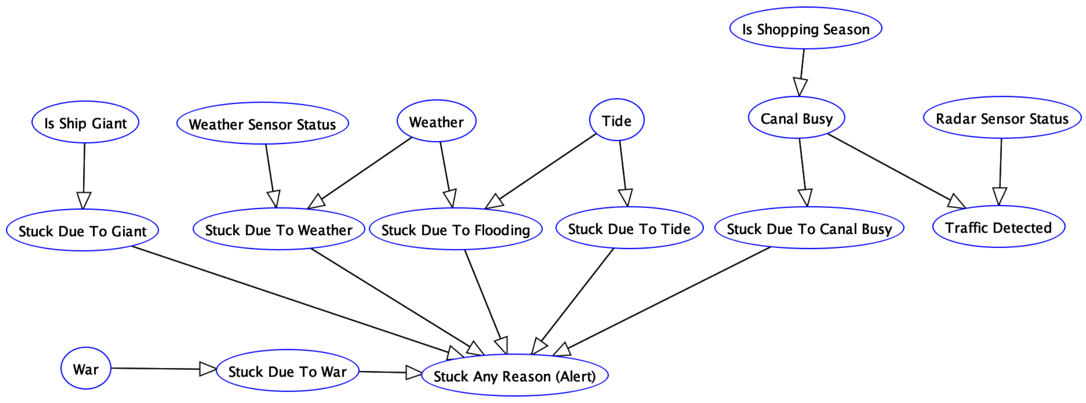


Figure 18: The top and middle screenshots show the `CANAL1.xml` and `CANAL2.xml` Bayesian networks respectively. The bottom screenshot shows the program result of merging these two networks. The result is as expected and was verified further for correctness (i.e., by checking the probability tables). Note that, as `CANAL2.xml` is simply an extension of `CANAL1.xml`, the result of merging these two networks is simply `CANAL2.xml`. This is because all variables are in the intersection, so all variables are internal and trigger case C when merging. Case C takes the BN2 variables by default when a variable shares the same number of parents in both networks being merged, which is the case for every variable here.





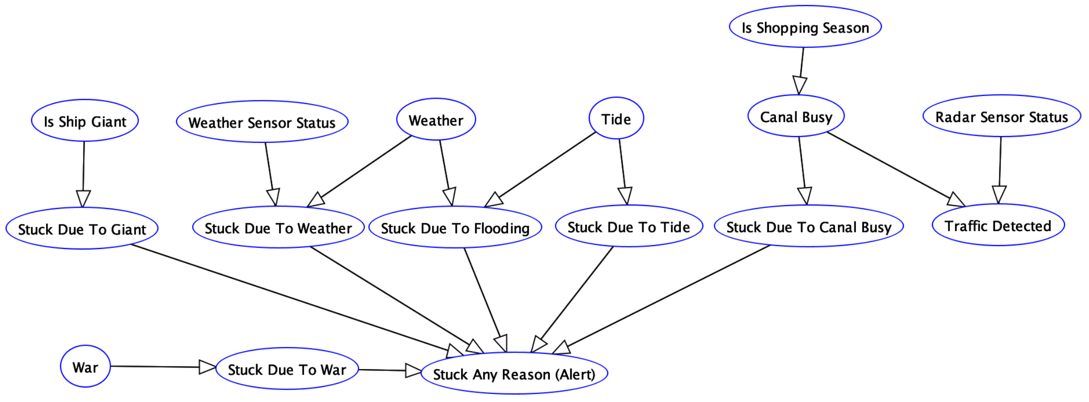
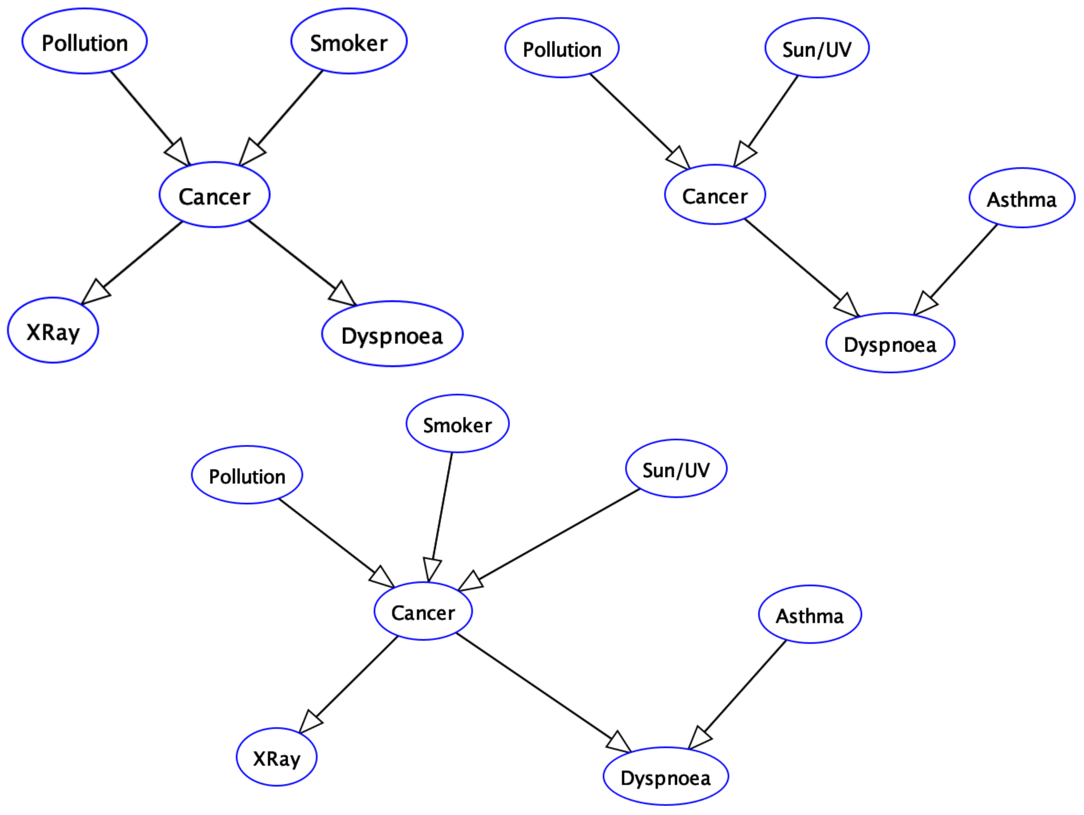


Figure 19: The top screenshots show the `MEDICAL1.xml` and `MEDICAL2.xml` Bayesian networks respectively, which were sourced from the lecture material. The bottom screenshot shows the program result of merging these two networks. The result is as expected and was verified further for correctness (i.e., by checking the probability tables).



B.2.3 Part 2 - Discussion

A successful implementation of the merging algorithm presented in the lecture material has been developed. The developed program correctly merges the pairs of Bayesian networks provided with this assessment. However, the implementation is limited; it may not correctly merge Bayesian networks in general.

The implementation of merging conditional probability tables for external variables has not been generalised (i.e., cannot handle any combination of conditional probability tables, specifically, where the conditional probabilities have some number of events in common). A more robust representation of conditional probability tables in the variable data type (instead of closely matching the format of the `XML` files) would make this trivial to solve, but would also require significant reworking of the current implementation. Despite this limitation, the specification only requires the implementation to work for the three pairs of BNs formed with the assessment, so the implementation is acceptable.

Also, whilst the implementation improves on the specification by disregarding the assumption that variables must have a maximum of 5 parents, the program is limited by the general assumptions made by the merging algorithm. Specifically, future work should investigate the merging of Bayesian networks where variables have a non-Boolean domain, or where the network graph contains cycles, which are common occurrences in practice.

C Part 3 Extension

C.1 Problem Description And Modelling

An additional problem is specified and modelled. The problem concerns the identification of fraudulent transactions by a credit card company.

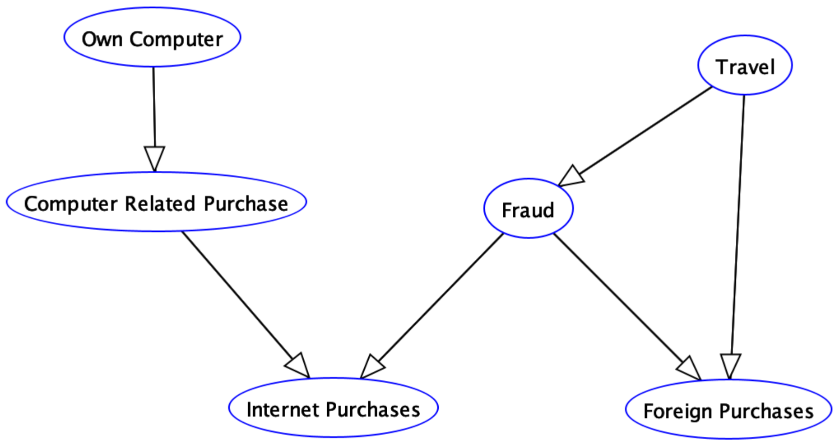
Let 5% of transactions occur whilst a card holder is travelling, on average. Travelling increases the likelihood of fraudulent transactions as tourists are ideal targets for thieves. Let 1% of transactions be fraudulent when a card holder is travelling, versus only 0.2% when not travelling.

If a transaction is fraudulent, the likelihood of there being a foreign purchase increases, unless the card holder happens to be travelling. When a card holder is not travelling, let 10% of fraudulent transactions and only 1% of legitimate transactions be foreign purchases. When a card holder is travelling, let 90% of transactions be foreign purchases regardless of transaction legitimacy.

Additionally, internet purchases are more likely to be fraudulent. This is especially true when the card holder does not own a computer. Let 90% of card holders own a computer; 1% of their legitimate transactions and 2% of their fraudulent transactions are internet purchases. Ownership of a computer is not known to the credit card company but can be guessed from the transaction history based on computer-related purchases; 10% of those who own a computer purchase some computer related item in a given week, as opposed to just 0.01% of those who don't own any computer.

The developed model for this scenario is seen in figure 20. The model has been provided as '`PART3_CREDIT.xml`'; it can be verified that the conditional probability tables of this BN are as expected.

Figure 20: The Bayesian network developed for the Part 3 problem description, `PART3_CREDIT.xml`.



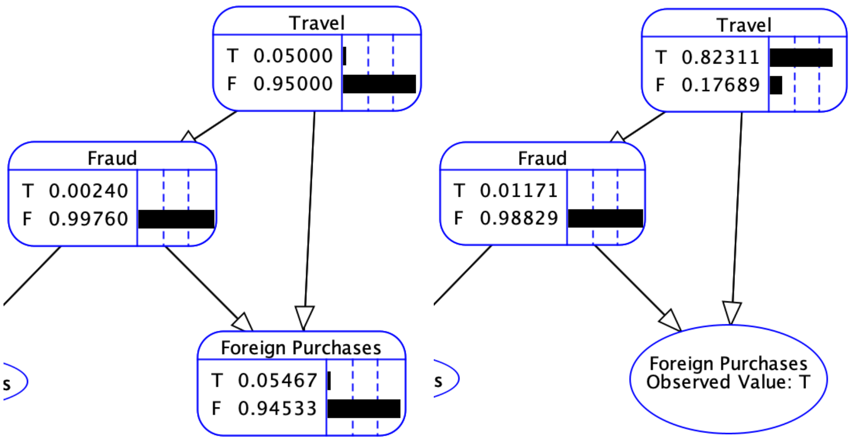
One criticism of this model is that it is oversimplified; simplicity is desired in a model but leads to undesirable relationships in this case. For example, observing that `Internet Purchases`=*True*, (minorly) increases the likelihood that the card holder is travelling. One can argue the opposite relationship is true; internet purchases are more likely to be made from home, and in-person purchases are more likely when travelling.

C.2 Predictive And Diagnostic Queries

Diagnostic Query - Given that a transaction is a foreign purchase, what is the likelihood the transaction is fraudulent, and how does this affect the probability that the card holder is travelling?

- Given that a transaction is a foreign purchase, the likelihood that the transaction is fraudulent increases from ~ 0.002 (before observation) to ~ 0.01. Also, the likelihood that the card holder is travelling increases from 0.05 (before observation) to ~ 0.82. These results are expected; foreign purchases are more likely to have been caused by a travelling card holder, who is more likely to have had there credit card stolen as a result.

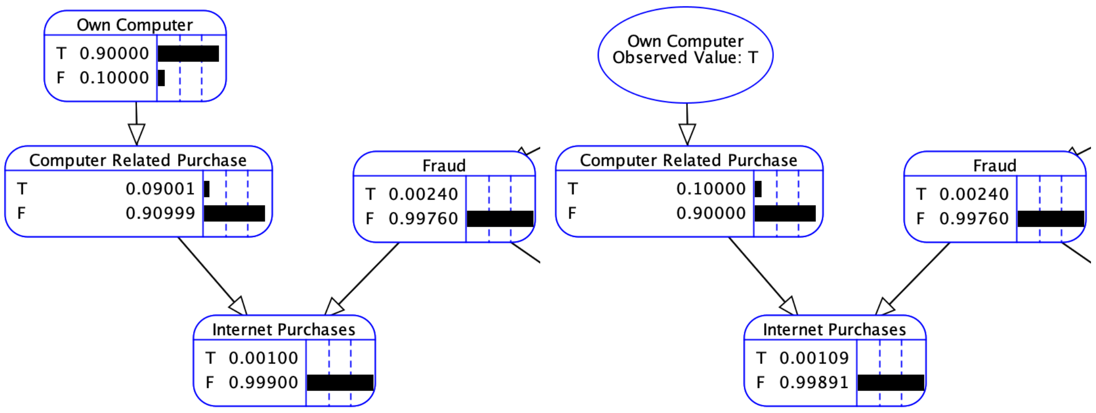
Figure 21: .



Predictive Query - Observing that a card holder owns a computer, what is the likelihood that their transaction history has computer-related purchases (so the credit card company can guess their computer ownership), and therefore that a transaction is an internet purchase?

- Given that a card holder owns a computer, the network predicts that their transaction history will have a computer-related purchase with 10% likelihood (up from $\sim$ 9% before the evidence). Also, it is predicted that a transaction will therefore be an internet purchase with $\sim$ 0.11% likelihood (up from $\sim$ 0.10% before the evidence). Thus, computer ownership makes card holders more likely to purchase computer-related goods and make internet purchases, as expected by the reasoning behind the network's creation.

Figure 22: .



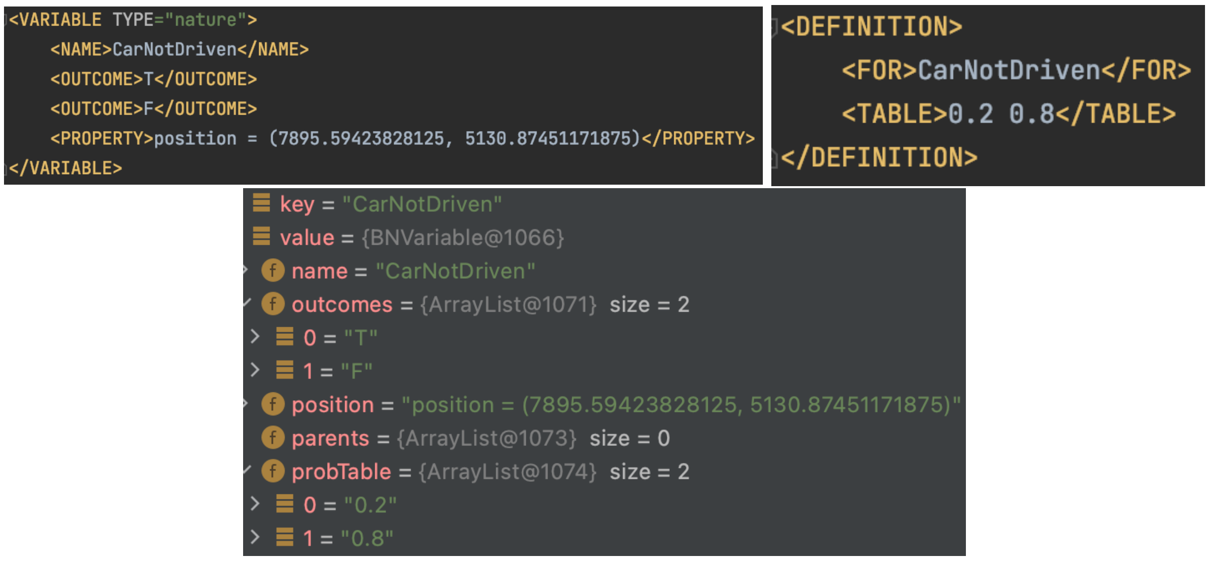
D Testing Summary

D.1 Part 2

Table 2: Table outlining the core correctness testing carried out for the Part 2 `A4main` program.

Test Description	Passed?	Proof
Test a variable with no parents is correctly read from `XML` file.	Yes	Figure 23
Test a variable with one parent is correctly read from `XML` file.	Yes	Figure 24
Test a variable with multiple parent is correctly read from `XML` file.	Yes	Figure 25
Test a Bayesian network is correctly read from `XML` file.	Yes	Figure 26
Test non-intersection, internal, and external variables are correctly identified.	Yes	Figure 27
Test non-intersection variables are correctly handled by the program.	Yes	Figure 28
Test internal variables are correctly handled by the program.	Yes	Figure 29
Test external variables are correctly handled by the program.	Yes	Figure 30
Test merging conditional probability tables (for external variables) is correct.	Yes	Figure 31
Test the merged BN is wrote to an `XML` file and is readable by `bayes.jar`.	Yes	Figure 32

Figure 23: Test showing that a variable which has no parents is correctly parsed from the BN `XML` file. Here, the variable *CarNotDriven* from `example1.xml` has been parsed correctly; its attributes (top) are correctly seen in memory (bottom) and no parents are recorded.

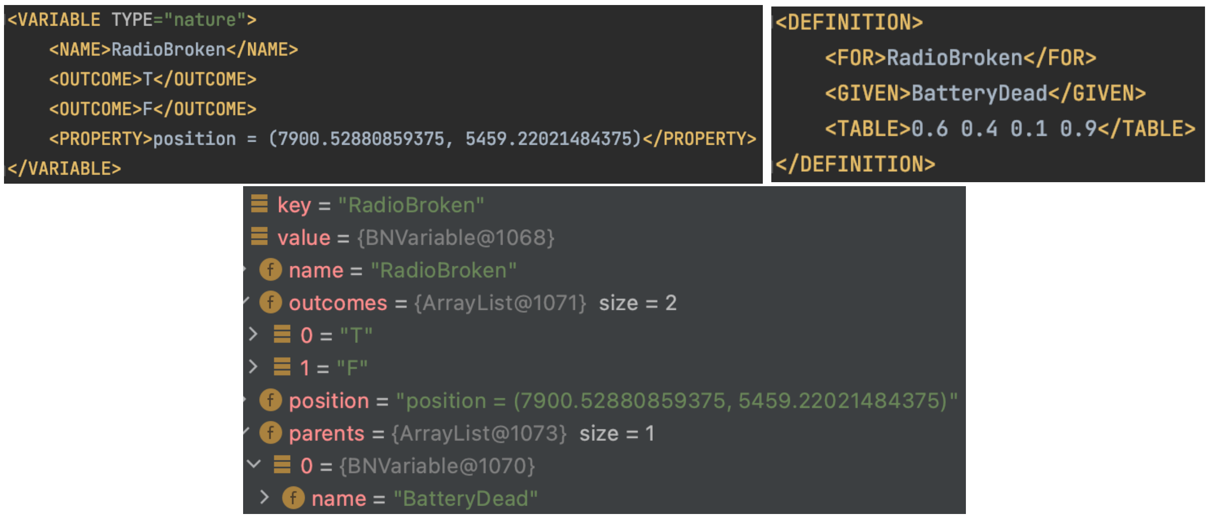


```
<VARIABLE TYPE="nature">
    <NAME>CarNotDriven</NAME>
    <OUTCOME>T</OUTCOME>
    <OUTCOME>F</OUTCOME>
    <PROPERTY>position = (7895.59423828125, 5130.87451171875)</PROPERTY>
</VARIABLE>
```

```
<DEFINITION>
    <FOR>CarNotDriven</FOR>
    <TABLE>0.2 0.8</TABLE>
</DEFINITION>
```

```
key = "CarNotDriven"
value = {BNVariable@1066}
 name = "CarNotDriven"
 outcomes = {ArrayList@1071}  size = 2
  0 = "T"
  1 = "F"
 position = "position = (7895.59423828125, 5130.87451171875)"
 parents = {ArrayList@1073}  size = 0
 probTable = {ArrayList@1074}  size = 2
  0 = "0.2"
  1 = "0.8"
```

Figure 24: Test showing that a variable which has a single parent is correctly parsed from the BN `XML` file. Here, the variable *RadioBroken* from `example1.xml` has been parsed correctly. Its attributes (top) are correctly seen in memory (bottom), and its sole parent, *BatteryDead*, is recorded.

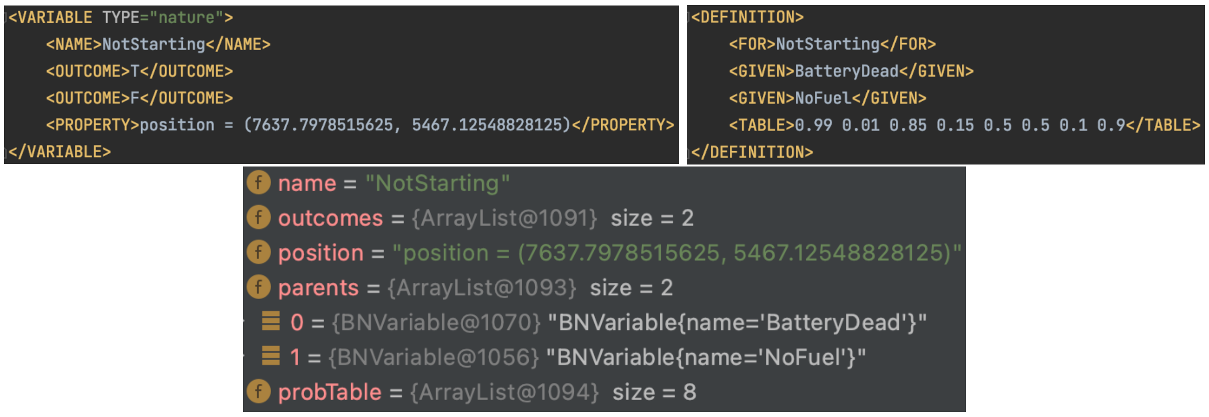


```
<VARIABLE TYPE="nature">
    <NAME>RadioBroken</NAME>
    <OUTCOME>T</OUTCOME>
    <OUTCOME>F</OUTCOME>
    <PROPERTY>position = (7900.52880859375, 5459.22021484375)</PROPERTY>
</VARIABLE>
```

```
<DEFINITION>
    <FOR>RadioBroken</FOR>
    <GIVEN>BatteryDead</GIVEN>
    <TABLE>0.6 0.4 0.1 0.9</TABLE>
</DEFINITION>
```

```
key = "RadioBroken"
value = {BNVariable@1068}
 name = "RadioBroken"
 outcomes = {ArrayList@1071}  size = 2
>  0 = "T"
>  1 = "F"
 position = "position = (7900.52880859375, 5459.22021484375)"
 parents = {ArrayList@1073}  size = 1
v  0 = {BNVariable@1070}
>   name = "BatteryDead"
```

Figure 25: Test showing that a variable which has multiple parents is correctly parsed from the BN `XML` file. Here, the variable *NotStarting* from `example1.xml` has been parsed correctly. Its attributes (top) are correctly seen in memory (bottom), and its parents, *BatteryDead* and *NoFuel*, are recorded.

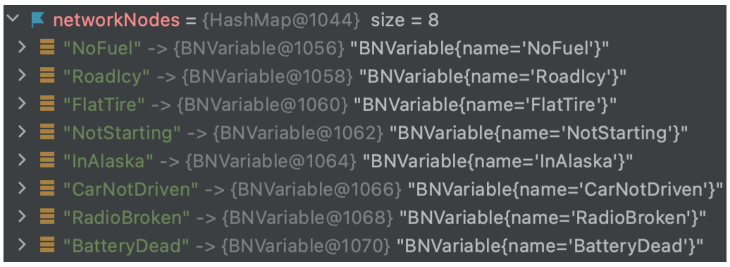


```
<VARIABLE TYPE="nature">
    <NAME>NotStarting</NAME>
    <OUTCOME>T</OUTCOME>
    <OUTCOME>F</OUTCOME>
    <PROPERTY>position = (7637.7978515625, 5467.12548828125)</PROPERTY>
</VARIABLE>
```

```
<DEFINITION>
    <FOR>NotStarting</FOR>
    <GIVEN>BatteryDead</GIVEN>
    <GIVEN>NoFuel</GIVEN>
    <TABLE>0.99 0.01 0.85 0.15 0.5 0.5 0.1 0.9</TABLE>
</DEFINITION>
```

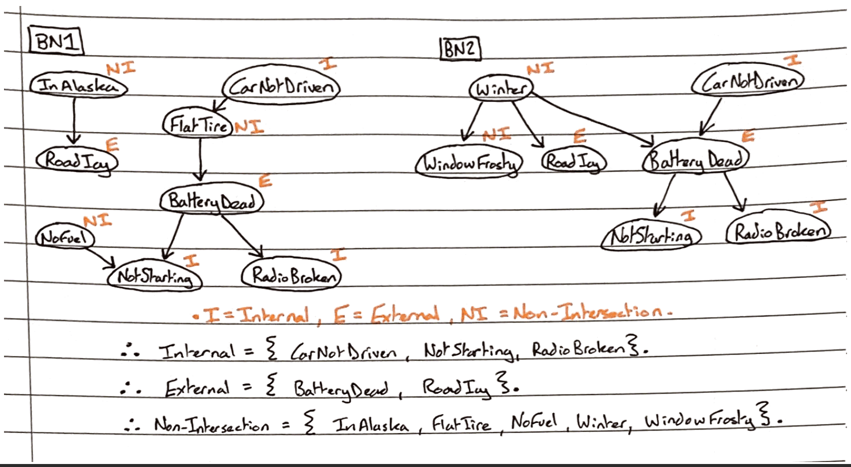
```
name = "NotStarting"
outcomes = {ArrayList@1091}  size = 2
position = "position = (7637.7978515625, 5467.12548828125)"
parents = {ArrayList@1093}  size = 2
 0 = {BNVariable@1070} "BNVariable{name='BatteryDead'}"
 1 = {BNVariable@1056} "BNVariable{name='NoFuel'}"
probTable = {ArrayList@1094}  size = 8
```

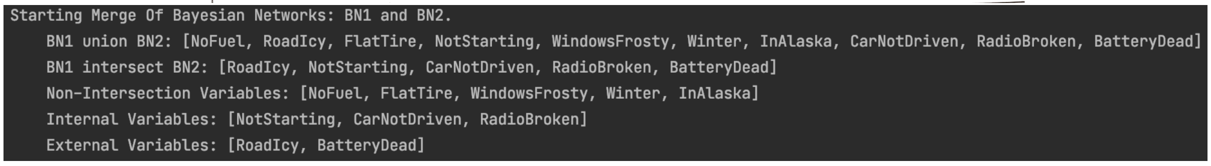
Figure 26: Test showing that a Bayesian network, `example1.xml`, is correctly parsed from a BN `XML` file. It has been tested that each variable of a network is parsed correctly, so simply checking the BN program object has all of the expected variables shows that BNs are parsed correctly. The `example1.xml` BN has the following variables: *FlatTire*, *RoadIcy*, *BatteryDead*, *RadioBroken*, *CarNotDriven*, *NotStarting*, *InAlaska*, and *NoFuel*. These are all seen in the program output below.



```
v networkNodes = {HashMap@1044}  size = 8
 >  "NoFuel" -> {BNVariable@1056} "BNVariable{name='NoFuel'}"
 >  "RoadIcy" -> {BNVariable@1058} "BNVariable{name='RoadIcy'}"
 >  "FlatTire" -> {BNVariable@1060} "BNVariable{name='FlatTire'}"
 >  "NotStarting" -> {BNVariable@1062} "BNVariable{name='NotStarting'}"
 >  "InAlaska" -> {BNVariable@1064} "BNVariable{name='InAlaska'}"
 >  "CarNotDriven" -> {BNVariable@1066} "BNVariable{name='CarNotDriven'}"
 >  "RadioBroken" -> {BNVariable@1068} "BNVariable{name='RadioBroken'}"
 >  "BatteryDead" -> {BNVariable@1070} "BNVariable{name='BatteryDead'}"
```

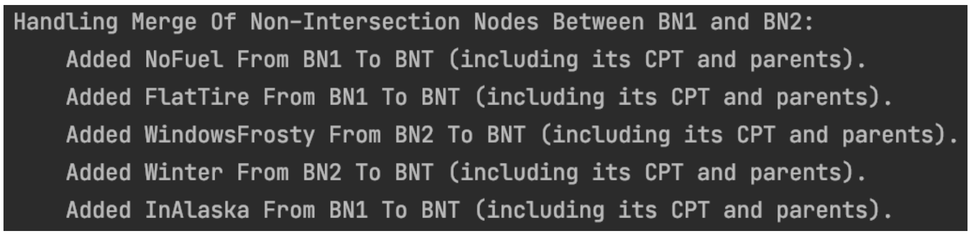
Figure 27: Tests showing that the sets of non-intersection, internal, and external variables are identified correctly. In this case, `example1.xml` and `example2.xml` are being merged. The correct variable sets were manually calculated (top) and the same sets are produced by the program (bottom).


BN1
InAlaska
NI
CarNotDriven
I
FlatTire
NI
RoadIcy
E
BatteryDead
E
NoFuel
NI
NotStarting
I
RadioBroken
I
BN2
Winter
NI
CarNotDriven
I
WindowFrosty
NI
RoadIcy
E
BatteryDead
E
NotStarting
I
RadioBroken
I
• I = Internal , E = External , NI = Non-Intersection.
∴ Internal = { CarNotDriven , NotStarting, RadioBroken }.
∴ External = { BatteryDead , RoadIcy }.
∴ Non-Intersection = { InAlaska , FlatTire , NoFuel , Winter, WindowFrosty }.



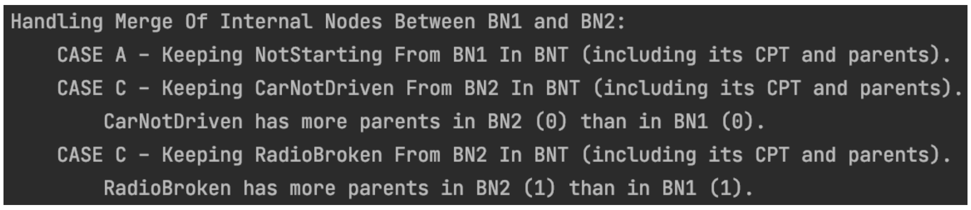
```
Starting Merge Of Bayesian Networks: BN1 and BN2.
    BN1 union BN2: [NoFuel, RoadIcy, FlatTire, NotStarting, WindowsFrosty, Winter, InAlaska, CarNotDriven, RadioBroken, BatteryDead]
    BN1 intersect BN2: [RoadIcy, NotStarting, CarNotDriven, RadioBroken, BatteryDead]
    Non-Intersection Variables: [NoFuel, FlatTire, WindowsFrosty, Winter, InAlaska]
    Internal Variables: [NotStarting, CarNotDriven, RadioBroken]
    External Variables: [RoadIcy, BatteryDead]
```

Figure 28: Test showing that non-intersection variables are handled correctly by the merge. For each variable in the non-intersection, all of its dependencies and its conditional probability table from its previous network are included in the merged network. This is seen in the following screenshot, where the non-intersection variables identified in figure 27 have been added to the merged network appropriately, as required.



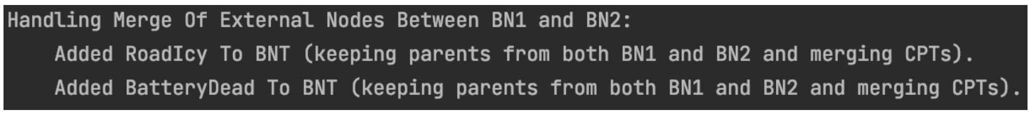
```
Handling Merge Of Non-Intersection Nodes Between BN1 and BN2:
    Added NoFuel From BN1 To BNT (including its CPT and parents).
    Added FlatTire From BN1 To BNT (including its CPT and parents).
    Added WindowsFrosty From BN2 To BNT (including its CPT and parents).
    Added Winter From BN2 To BNT (including its CPT and parents).
    Added InAlaska From BN1 To BNT (including its CPT and parents).
```

Figure 29: Test showing the internal variables are handled correctly by the merge. For each variable in the internal variables set, the variable is handled by the *DELETE* rule. The delete rule has three cases: A, B, and C. For the example with `example1.xml` and `example2.xml`, the following should occur: *NotStarting* is case A as it has a parent in BN1 not in the intersection, *CarNotDriven* is case C as its parents (the empty set) are in the intersection, and *RadioBroken* is case C as its parents (*BatteryDead*) are in the intersection. Note that, for case C, if the number of parents in each network are the same, then the variable in BN2 is used by default. These requirements are seen to be the case in the following program output.



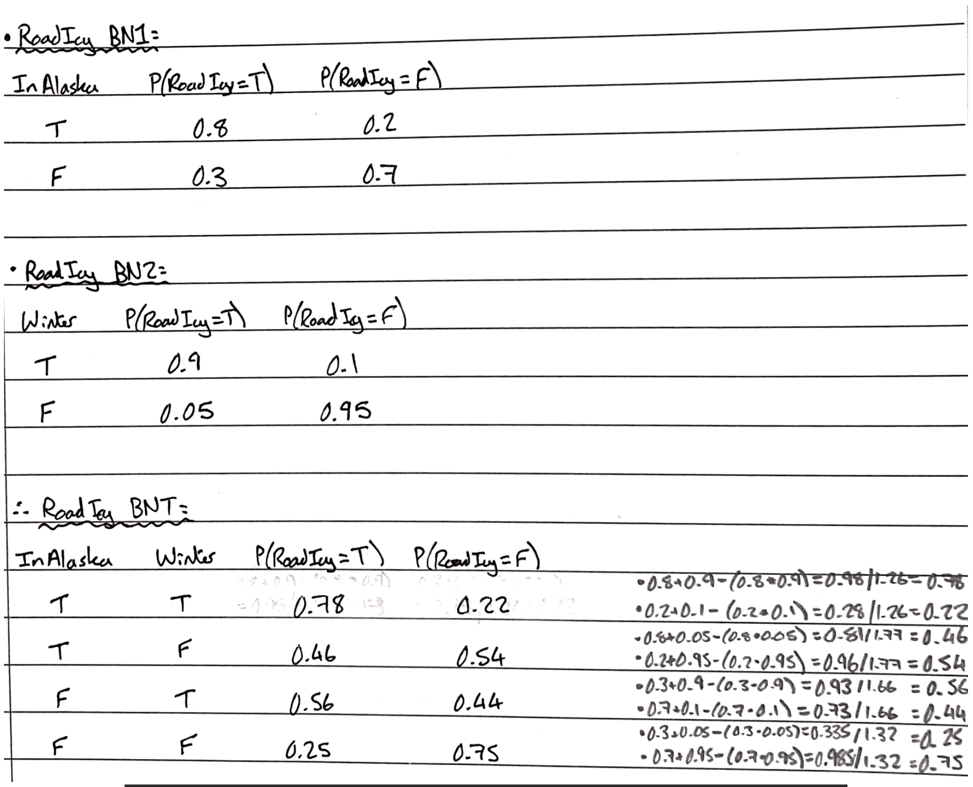
```
Handling Merge Of Internal Nodes Between BN1 and BN2:
    CASE A - Keeping NotStarting From BN1 In BNT (including its CPT and parents).
    CASE C - Keeping CarNotDriven From BN2 In BNT (including its CPT and parents).
        CarNotDriven has more parents in BN2 (0) than in BN1 (0).
    CASE C - Keeping RadioBroken From BN2 In BNT (including its CPT and parents).
        RadioBroken has more parents in BN2 (1) than in BN1 (1).
```

Figure 30: Test showing the external variables are handled correctly by the merge. For each variable in the external variables set, a combine step is used. The combine step adds the variable to the merged network along with all of its parents from both networks and merges the conditional probability tables of the variable from both networks. These steps are shown to occur by the program when merging `example1.xml` and `example2.xml`. The external variables, *RoadIcy* and *BatteryDead*, are added to the merged network with all of its parents and merged probability table.



```
Handling Merge Of External Nodes Between BN1 and BN2:
    Added RoadIcy To BNT (keeping parents from both BN1 and BN2 and merging CPTs).
    Added BatteryDead To BNT (keeping parents from both BN1 and BN2 and merging CPTs).
```

Figure 31: Test showing that the conditional probability tables of a variable from both networks are correctly merged. In the case of merging `example1.xml` and `example2.xml`, the *RoadIcy* variable has been used as an example. The top screenshot shows the expected merged conditional probability table for this variable, and the bottom screenshot shows that this is correctly produced by the program (to more significant figures). Similar testing was carried out for variables with different amounts of parents in either network.



• RoadIcy BN1:

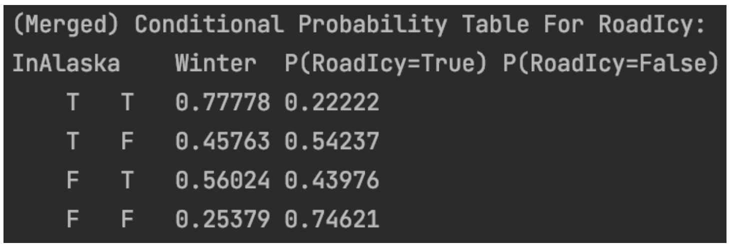
InAlaska	P(RoadIcy=T)	P(RoadIcy=F)
T	0.8	0.2
F	0.3	0.7

• RoadIcy BN2:

Winter	P(RoadIcy=T)	P(RoadIcy=F)
T	0.9	0.1
F	0.05	0.95

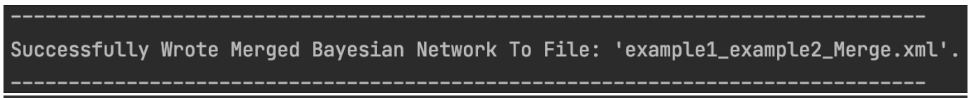
∴ RoadIcy BNT:

InAlaska	Winter	P(RoadIcy=T)	P(RoadIcy=F)	
T	T	0.78	0.22	• 0.8+0.9−(0.8∗0.9)=0.98/1.26=0.78 • 0.2+0.1−(0.2∗0.1)=0.28/1.26=0.22
T	F	0.46	0.54	• 0.8+0.05−(0.8∗0.05)=0.81/1.77=0.46 • 0.2+0.95−(0.2·0.95)=0.96/1.77=0.54
F	T	0.56	0.44	• 0.3+0.9−(0.3·0.9)=0.93/1.66=0.56 • 0.7+0.1−(0.7·0.1)=0.73/1.66=0.44
F	F	0.25	0.75	• 0.3+0.05−(0.3·0.05)=0.335/1.32=0.25 • 0.7+0.95−(0.7·0.95)=0.985/1.32=0.75

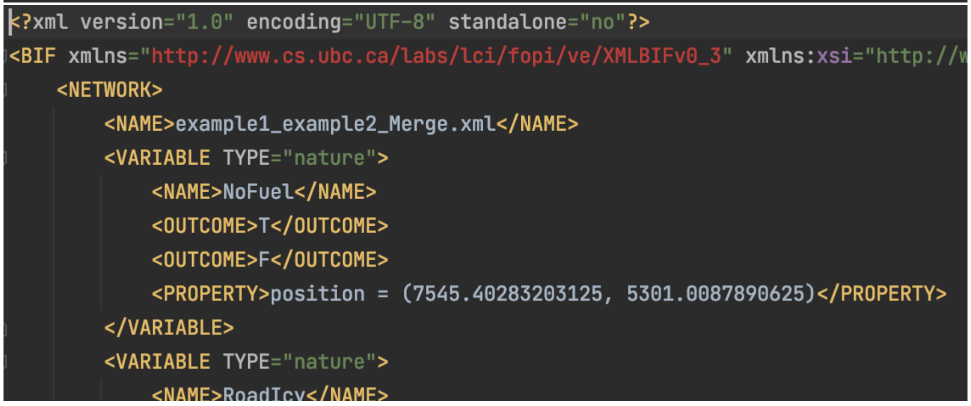


```
(Merged) Conditional Probability Table For RoadIcy:
InAlaska    Winter  P(RoadIcy=True) P(RoadIcy=False)
    T   T   0.77778 0.22222
    T   F   0.45763 0.54237
    F   T   0.56024 0.43976
    F   F   0.25379 0.74621
```

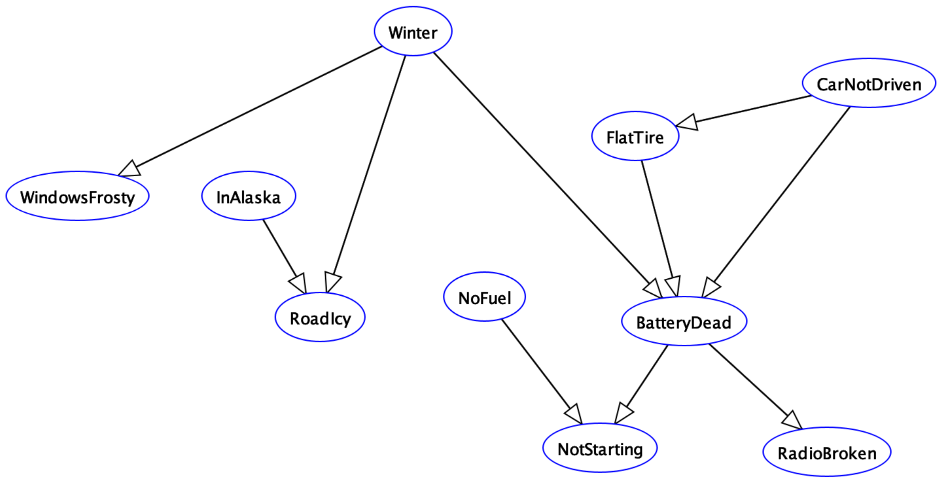
Figure 32: Test showing the merged Bayesian network is correctly wrote to an XML file and that said file is readable by `bayes.jar`. The top screenshot shows that the program indicates an XML file for the merged BN has been written. The middle screenshot shows that this file is in fact of the correct `XML` format required. The bottom screenshot shows that the merged BN is therefore view-able in the `bayes.jar` tool, as required. It was also verified that the probability tables shown in `bayes.jar` are consistent with those produced by the program for the merged BN.



```
------------------------------------------------------------------------------
Successfully Wrote Merged Bayesian Network To File: 'example1_example2_Merge.xml'.
------------------------------------------------------------------------------
```



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<BIF xmlns="http://www.cs.ubc.ca/labs/lci/fopi/ve/XMLBIFv0_3" xmlns:xsi="http://w
    <NETWORK>
        <NAME>example1_example2_Merge.xml</NAME>
        <VARIABLE TYPE="nature">
            <NAME>NoFuel</NAME>
            <OUTCOME>T</OUTCOME>
            <OUTCOME>F</OUTCOME>
            <PROPERTY>position = (7545.40283203125, 5301.0087890625)</PROPERTY>
        </VARIABLE>
        <VARIABLE TYPE="nature">
            <NAME>RoadIcy</NAME>
```



E Bibliography

The required content for this assessment was acquired from the specification and module material.

F Appendix

The program output of the merging algorithm, for each of the pairs of BNs used in this assessment, are shown in this section as requested by the specification.

F.1 Program Output - `example1.xml` and `example2.xml` Merge

```
Successfully Read Bayesian Network From File: 'example1.xml'.
------------------------------------------------------------------------------
Successfully Read Bayesian Network From File: 'example2.xml'.
------------------------------------------------------------------------------
Starting Merge Of Bayesian Networks: BN1 and BN2.
    BN1 union BN2: [NoFuel, RoadIcy, FlatTire, NotStarting, WindowsFrosty, Winter,
    InAlaska, CarNotDriven, RadioBroken, BatteryDead]
    BN1 intersect BN2: [RoadIcy, NotStarting, CarNotDriven, RadioBroken, BatteryDead]
    Non-Intersection Variables: [NoFuel, FlatTire, WindowsFrosty, Winter, InAlaska]
    Internal Variables: [NotStarting, CarNotDriven, RadioBroken]
    External Variables: [RoadIcy, BatteryDead]
Handling Merge Of Non-Intersection Nodes Between BN1 and BN2:
    Added NoFuel From BN1 To BNT (including its CPT and parents).
    Added FlatTire From BN1 To BNT (including its CPT and parents).
    Added WindowsFrosty From BN2 To BNT (including its CPT and parents).
    Added Winter From BN2 To BNT (including its CPT and parents).
    Added InAlaska From BN1 To BNT (including its CPT and parents).
Handling Merge Of Internal Nodes Between BN1 and BN2:
    CASE A - Keeping NotStarting From BN1 In BNT (including its CPT and parents).
    CASE C - Keeping CarNotDriven From BN2 In BNT (including its CPT and parents).
        CarNotDriven has more parents in BN2 (0) than in BN1 (0).
    CASE C - Keeping RadioBroken From BN2 In BNT (including its CPT and parents).
        RadioBroken has more parents in BN2 (1) than in BN1 (1).
Handling Merge Of External Nodes Between BN1 and BN2:
    Added RoadIcy To BNT (keeping parents from both BN1 and BN2 and merging CPTs).
        (Merged) Conditional Probability Table For RoadIcy:
        Winter InAlaska P(RoadIcy=True) P(RoadIcy=False)
          T       T       0.77778       0.22222
          T       F       0.45763       0.54237
          F       T       0.56024       0.43976
          F       F       0.25379       0.74621
    Added BatteryDead To BNT (keeping parents from both BN1 and BN2 and merging CPTs).
        (Merged) Conditional Probability Table For BatteryDead:
        FlatTire Winter CarNotDriven P(BatteryDead=True) P(BatteryDead=False)
          T        T           T        0.62203            0.37797
          T        T           F        0.59494            0.40506
          T        F           T        0.53247            0.46753
          T        F           F        0.28575            0.71425
          F        T           T        0.49774            0.50226
          F        T           F        0.47394            0.52606
          F        F           T        0.41204            0.58796
          F        F           F        0.00110            0.99890
------------------------------------------------------------------------------
Final Merged Network: example1_example2_Merge.xml
    Variable: NoFuel
        Outcomes: [T, F]
        Parents: []
        (Merged) Conditional Probability Table For NoFuel:
        P(NoFuel=True) P(NoFuel=False)
           0.15000        0.85000
    Variable: RoadIcy
        Outcomes: [T, F]
        Parents: [Winter, InAlaska]
        (Merged) Conditional Probability Table For RoadIcy:
        Winter InAlaska P(RoadIcy=True) P(RoadIcy=False)
          T       T         0.77778          0.22222
          T       F         0.45763          0.54237
          F       T         0.56024          0.43976
          F       F         0.25379          0.74621
    Variable: FlatTire
        Outcomes: [T, F]
        Parents: [CarNotDriven]
        (Merged) Conditional Probability Table For FlatTire:
        CarNotDriven P(FlatTire=True) P(FlatTire=False)
            T            0.70000           0.30000
            F            0.20000           0.80000
    Variable: WindowsFrosty
        Outcomes: [T, F]
        Parents: [Winter]
        (Merged) Conditional Probability Table For WindowsFrosty:
        Winter P(WindowsFrosty=True) P(WindowsFrosty=False)
          T        0.30000                   0.70000
          F        0.00100                   0.99900
    Variable: NotStarting
        Outcomes: [T, F]
        Parents: [BatteryDead, NoFuel]
        (Merged) Conditional Probability Table For NotStarting:
        BatteryDead NoFuel P(NotStarting=True) P(NotStarting=False)
            T        T        0.99000            0.01000
            T        F        0.85000            0.15000
            F        T        0.50000            0.50000
            F        F        0.10000            0.90000
    Variable: Winter
        Outcomes: [T, F]
        Parents: []
        (Merged) Conditional Probability Table For Winter:
        P(Winter=True) P(Winter=False)
            0.05000        0.95000
    Variable: InAlaska
        Outcomes: [T, F]
        Parents: []
        (Merged) Conditional Probability Table For InAlaska:
        P(InAlaska=True) P(InAlaska=False)
            0.30000            0.70000
    Variable: CarNotDriven
        Outcomes: [T, F]
        Parents: []
        (Merged) Conditional Probability Table For CarNotDriven:
        P(CarNotDriven=True) P(CarNotDriven=False)
            0.20000                  0.80000
    Variable: RadioBroken
        Outcomes: [T, F]
        Parents: [BatteryDead]
        (Merged) Conditional Probability Table For RadioBroken:
        BatteryDead P(RadioBroken=True) P(RadioBroken=False)
            T        0.60000             0.40000
            F        0.00100             0.99900
    Variable: BatteryDead
        Outcomes: [T, F]
```

```
        Parents: [FlatTire, Winter, CarNotDriven]
        (Merged) Conditional Probability Table For BatteryDead:
        FlatTire Winter CarNotDriven P(BatteryDead=True) P(BatteryDead=False)
          T          T         T           0.62203         0.37797
          T          T         F           0.59494         0.40506
          T          F         T           0.53247         0.46753
          T          F         F           0.28575         0.71425
          F          T         T           0.49774         0.50226
          F          T         F           0.47394         0.52606
          F          F         T           0.41204         0.58796
          F          F         F           0.00110         0.99890
--------------------------------------------------------------------------------
Successfully Wrote Merged Bayesian Network To File: 'example1_example2_Merge.xml'.
--------------------------------------------------------------------------------
```

F.2 Program Output - `CANAL1.xml` and `CANAL2.xml` Merge

```
Successfully Read Bayesian Network From File: 'CANAL1.xml'.
-------------------------------------------------------------------------------
Successfully Read Bayesian Network From File: 'CANAL2.xml'.
-------------------------------------------------------------------------------
Starting Merge Of Bayesian Networks: BN1 and BN2.
    BN1 union BN2: [Stuck Due To Weather, War, Traffic Detected, Canal Busy,
    Stuck Due To Canal Busy, Stuck Due To Tide, Stuck Due To Flooding, Is Ship Giant,
    Weather, Tide, Stuck Due To War, Stuck Due To Giant, Radar Sensor Status,
    Weather Sensor  Status, Stuck Any Reason (Alert), Is Shopping Season]
    BN1 intersect BN2: [Stuck Due To Weather, Traffic Detected, Canal Busy,
    Stuck Due To Canal Busy, Stuck Due To Tide, Is Ship Giant, Weather, Tide,
    Stuck Due To Giant, Radar Sensor Status, Weather Sensor Status,
    Stuck Any Reason (Alert), Is Shopping Season]
    Non-Intersection Variables: [War, Stuck Due To Flooding, Stuck Due To War]
    Internal Variables: [Stuck Due To Weather, Traffic Detected, Canal Busy,
    Stuck Due To Canal Busy, Stuck Due To Tide, Is Ship Giant, Weather, Tide,
    Stuck Due To Giant, Radar Sensor Status, Weather Sensor Status,
    Stuck Any Reason (Alert), Is Shopping Season]
    External Variables: []
Handling Merge Of Non-Intersection Nodes Between BN1 and BN2:
    Added War From BN2 To BNT (including its CPT and parents).
    Added Stuck Due To Flooding From BN2 To BNT (including its CPT and parents).
    Added Stuck Due To War From BN2 To BNT (including its CPT and parents).
Handling Merge Of Internal Nodes Between BN1 and BN2:
    CASE C - Keeping Stuck Due To Weather From BN2 In BNT (including its CPT and parents).
        Stuck Due To Weather has more parents in BN2 (2) than in BN1 (2).
    CASE C - Keeping Traffic Detected From BN2 In BNT (including its CPT and parents).
        Traffic Detected has more parents in BN2 (2) than in BN1 (2).
    CASE C - Keeping Canal Busy From BN2 In BNT (including its CPT and parents).
        Canal Busy has more parents in BN2 (1) than in BN1 (1).
    CASE C - Keeping Stuck Due To Canal Busy From BN2 In BNT (including its CPT and parents).
        Stuck Due To Canal Busy has more parents in BN2 (1) than in BN1 (1).
    CASE C - Keeping Stuck Due To Tide From BN2 In BNT (including its CPT and parents).
        Stuck Due To Tide has more parents in BN2 (1) than in BN1 (1).
    CASE C - Keeping Is Ship Giant From BN2 In BNT (including its CPT and parents).
        Is Ship Giant has more parents in BN2 (0) than in BN1 (0).
    CASE C - Keeping Weather From BN2 In BNT (including its CPT and parents).
        Weather has more parents in BN2 (0) than in BN1 (0).
    CASE C - Keeping Tide From BN2 In BNT (including its CPT and parents).
        Tide has more parents in BN2 (0) than in BN1 (0).
    CASE C - Keeping Stuck Due To Giant From BN2 In BNT (including its CPT and parents).
        Stuck Due To Giant has more parents in BN2 (1) than in BN1 (1).
    CASE C - Keeping Radar Sensor Status From BN2 In BNT (including its CPT and parents).
        Radar Sensor Status has more parents in BN2 (0) than in BN1 (0).
    CASE C - Keeping Weather Sensor Status From BN2 In BNT (including its CPT and parents).
        Weather Sensor Status has more parents in BN2 (0) than in BN1 (0).
    CASE B - Keeping Stuck Any Reason (Alert) From BN2 In BNT (including its CPT and parents).
    CASE C - Keeping Is Shopping Season From BN2 In BNT (including its CPT and parents).
        Is Shopping Season has more parents in BN2 (0) than in BN1 (0).
Handling Merge Of External Nodes Between BN1 and BN2:
-------------------------------------------------------------------------------
Final Merged Network: CANAL1_CANAL2_Merge.xml
    Variable: War
        Outcomes: [Yes, No]
        Parents: []
        (Merged) Conditional Probability Table For War:
        P(War=True) P(War=False)
        0.00100     0.99900
Variable: Stuck Due To Weather
    Outcomes: [True, False]
    Parents: [Weather, Weather Sensor Status]
    (Merged) Conditional Probability Table For Stuck Due To Weather:
    Weather Weather Sensor Status P(Stuck Due To Weather=True) P(Stuck Due To Weather=False)
        T         T                    0.00700                      0.99300
        T         F                    0.01000                      0.99000
        F         T                    0.00400                      0.99600
        F         F                    0.00200                      0.99800
Variable: Traffic Detected
    Outcomes: [High, Low]
    Parents: [Canal Busy, Radar Sensor Status]
    (Merged) Conditional Probability Table For Traffic Detected:
    Canal Busy Radar Sensor Status P(Traffic Detected=True) P(Traffic Detected=False)
        T           T                  1.00000                      0.00000
        T           F                  0.50000                      0.50000
        F           T                  0.00000                      1.00000
        F           F                  0.50000                      0.50000
Variable: Canal Busy
    Outcomes: [Yes, No]
    Parents: [Is Shopping Season]
    (Merged) Conditional Probability Table For Canal Busy:
    Is Shopping Season P(Canal Busy=True) P(Canal Busy=False)
            T             0.90000             0.10000
            F             0.70000             0.30000
Variable: Stuck Due To Canal Busy
    Outcomes: [True, False]
    Parents: [Canal Busy]
    (Merged) Conditional Probability Table For Stuck Due To Canal Busy:
    Canal Busy P(Stuck Due To Canal Busy=True) P(Stuck Due To Canal Busy=False)
        T                 0.01000                          0.99000
        F                 0.00000                          1.00000
Variable: Stuck Due To Tide
    Outcomes: [True, False]
    Parents: [Tide]
    (Merged) Conditional Probability Table For Stuck Due To Tide:
    Tide P(Stuck Due To Tide=True) P(Stuck Due To Tide=False)
     T            0.01000                   0.99000
     F            0.00100                   0.99900
Variable: Stuck Due To Flooding
    Outcomes: [True, False]
    Parents: [Weather, Tide]
    (Merged) Conditional Probability Table For Stuck Due To Flooding:
    Weather Tide P(Stuck Due To Flooding=True) P(Stuck Due To Flooding=False)
      T      T              0.00050                                 0.99950
      T      F              0.00500                                 0.99500
      F      T              0.00010                                 0.99990
      F      F              0.00100                                 0.99900
Variable: Is Ship Giant
    Outcomes: [Yes, No]
    Parents: []
    (Merged) Conditional Probability Table For Is Ship Giant:
    P(Is Ship Giant=True) P(Is Ship Giant=False)
           0.75000                0.25000
Variable: Weather
    Outcomes: [Bad, Good]
    Parents: []
```

```
        (Merged) Conditional Probability Table For Weather:
        P(Weather=True) P(Weather=False)
            0.50000         0.50000
    Variable: Tide
        Outcomes: [Low, High]
        Parents: []
        (Merged) Conditional Probability Table For Tide:
        P(Tide=True) P(Tide=False)
           0.25000       0.75000
    Variable: Stuck Due To War
        Outcomes: [True, False]
        Parents: [War]
        (Merged) Conditional Probability Table For Stuck Due To War:
        War P(Stuck Due To War=True) P(Stuck Due To War=False)
         T          0.70000                    0.30000
         F          0.00000                    1.00000
    Variable: Stuck Due To Giant
        Outcomes: [True, False]
        Parents: [Is Ship Giant]
        (Merged) Conditional Probability Table For Stuck Due To Giant:
        Is Ship Giant P(Stuck Due To Giant=True) P(Stuck Due To Giant=False)
            T                 0.00500                      0.99500
            F                 0.00000                      1.00000
    Variable: Radar Sensor Status
        Outcomes: [Correct, Wrong]
        Parents: []
        (Merged) Conditional Probability Table For Radar Sensor Status:
        P(Radar Sensor Status=True) P(Radar Sensor Status=False)
                0.90000                         0.10000
    Variable: Weather Sensor Status
        Outcomes: [Correct, Wrong]
        Parents: []
        (Merged) Conditional Probability Table For Weather Sensor Status:
        P(Weather Sensor Status=True) P(Weather Sensor Status=False)
                    0.99000                         0.01000
    Variable: Stuck Any Reason (Alert)
        Outcomes: [True, False]
        Parents: [Stuck Due To Giant, Stuck Due To Weather, Stuck Due To Tide,
        Stuck Due To Canal Busy, Stuck Due To Flooding, Stuck Due To War]
        (Merged) Conditional Probability Table For Stuck Any Reason (Alert):
        Stuck Due To Giant Stuck Due To Weather Stuck Due To Tide Stuck Due To Canal Busy
        Stuck Due To Flooding Stuck Due To War    P(Stuck Any Reason (Alert)=True)
        P(Stuck Any Reason (Alert)=False)
            T T T T T T 1.00000 0.00000
            T T T T T F 1.00000 0.00000
                          ...
            F F F F F F 0.00100 0.99900
    Variable: Is Shopping Season
        Outcomes: [Yes, No]
        Parents: []
        (Merged) Conditional Probability Table For Is Shopping Season:
        P(Is Shopping Season=True) P(Is Shopping Season=False)
                0.20000                     0.80000
--------------------------------------------------------------------------------
Successfully Wrote Merged Bayesian Network To File: 'CANAL1_CANAL2_Merge.xml'.
--------------------------------------------------------------------------------
```

F.3 Program Output - `MEDICAL1.xml` and `MEDICAL2.xml` Merge

```
Successfully Read Bayesian Network From File: 'MEDICAL1.xml'.
--------------------------------------------------------------------------------
Successfully Read Bayesian Network From File: 'MEDICAL2.xml'.
--------------------------------------------------------------------------------
Starting Merge Of Bayesian Networks: BN1 and BN2.
BN1 union BN2: [Sun/UV, Pollution, Dyspnoea, Smoker, XRay, Cancer, Asthma]
    BN1 intersect BN2: [Pollution, Dyspnoea, Cancer]
    Non-Intersection Variables: [Sun/UV, Smoker, XRay, Asthma]
    Internal Variables: [Pollution, Dyspnoea]
    External Variables: [Cancer]
Handling Merge Of Non-Intersection Nodes Between BN1 and BN2:
    Added Sun/UV From BN2 To BNT (including its CPT and parents).
    Added Smoker From BN1 To BNT (including its CPT and parents).
    Added XRay From BN1 To BNT (including its CPT and parents).
    Added Asthma From BN2 To BNT (including its CPT and parents).
Handling Merge Of Internal Nodes Between BN1 and BN2:
    CASE C - Keeping Pollution From BN2 In BNT (including its CPT and parents).
        Pollution has more parents in BN2 (0) than in BN1 (0).
    CASE B - Keeping Dyspnoea From BN2 In BNT (including its CPT and parents).
Handling Merge Of External Nodes Between BN1 and BN2:
    Added Cancer To BNT (keeping parents from both BN1 and BN2 and merging CPTs).
        (Merged) Conditional Probability Table For Cancer:
        Sun/UV Pollution Smoker P(Cancer=True) P(Cancer=False)
          T        T       T        0.08103         0.91897
          T        T       F        0.08103         0.91897
          T        F       T        0.05593         0.94407
          T        F       F        0.05593         0.94407
          F        T       T        0.04710         0.95290
          F        T       F        0.04710         0.95290
          F        F       T        0.02055         0.97945
          F        F       F        0.02055         0.97945
--------------------------------------------------------------------------------
Final Merged Network: MEDICAL1_MEDICAL2_Merge.xml
    Variable: Sun/UV
        Outcomes: [T, F]
        Parents: []
        (Merged) Conditional Probability Table For Sun/UV:
        P(Sun/UV=True) P(Sun/UV=False)
            0.25000         0.75000
    Variable: Pollution
        Outcomes: [High, Low]
        Parents: []
        (Merged) Conditional Probability Table For Pollution:
        P(Pollution=True) P(Pollution=False)
            0.10000             0.90000
    Variable: Dyspnoea
        Outcomes: [T, F]
        Parents: [Cancer, Asthma]
        (Merged) Conditional Probability Table For Dyspnoea:
        Cancer Asthma P(Dyspnoea=True) P(Dyspnoea=False)
          T       T         0.95000            0.05000
          T       F         0.65000            0.35000
          F       T         0.60000            0.40000
          F       F         0.30000            0.70000
    Variable: Smoker
        Outcomes: [T, F]
        Parents: []
        (Merged) Conditional Probability Table For Smoker:
        P(Smoker=True) P(Smoker=False)
            0.30000         0.70000
    Variable: XRay
        Outcomes: [Positive, Negative]
        Parents: [Cancer]
        (Merged) Conditional Probability Table For XRay:
        Cancer P(XRay=True) P(XRay=False)
          T       0.90000         0.10000
          F       0.20000         0.80000
    Variable: Asthma
        Outcomes: [T, F]
        Parents: []
        (Merged) Conditional Probability Table For Asthma:
        P(Asthma=True) P(Asthma=False)
            0.08000         0.92000
    Variable: Cancer
        Outcomes: [T, F]
        Parents: [Sun/UV, Pollution, Smoker]
        (Merged) Conditional Probability Table For Cancer:
        Sun/UV Pollution Smoker P(Cancer=True) P(Cancer=False)
          T         T       T        0.08103         0.91897
          T         T       F        0.08103         0.91897
          T         F       T        0.05593         0.94407
          T         F       F        0.05593         0.94407
          F         T       T        0.04710         0.95290
          F         T       F        0.04710         0.95290
          F         F       T        0.02055         0.97945
          F         F       F        0.02055         0.97945
--------------------------------------------------------------------------------
Successfully Wrote Merged Bayesian Network To File: 'MEDICAL1_MEDICAL2_Merge.xml'.
--------------------------------------------------------------------------------
```