Assignment 1

MMS131 - Introduction to Artificial Intelligence
VT-23

Benjamin Elm Jonsson

Mechanical Engineering
Chalmers Institute of Technology
Gothenburg, Sweden

Contents

1 Problem 1

This problem consideres the *Wumpus world*.

1.1 a)

The first subproblem to this problem is to define all possible scenarios of a *Wumpus World*, where a stench, S, is percieved in either square $(1,2)$ or $(2,1)$. With the fact that Wumpuses, W, only occur in adjacent squares to where the stench is found the table can be constructed. All other facts regarding the *Wumpus world* are to be ignored.

Table 1: Table of all possible models

Index	$S_{1,2}$	$S_{2,1}$	$W_{1,3}$	$W_{2,2}$	$W_{3,1}$
1	F	F	F	F	F
2	F	F	F	F	T
3	F	F	F	T	F
4	F	F	F	T	T
5	F	F	T	F	F
6	F	F	T	F	T
7	F	F	T	T	F
8	F	F	T	T	T
9	F	T	F	F	F
10	F	T	F	F	T
11	F	T	F	T	F
12	F	T	F	T	T
13	F	T	T	F	F
14	F	T	T	F	T
15	F	T	T	T	F
16	F	T	T	T	T
17	T	F	F	F	F
18	T	F	F	F	T
19	T	F	F	T	F
21	T	F	F	T	T
21	T	F	T	F	F
22	T	F	T	F	T
23	T	F	T	T	F
24	T	F	T	T	T
25	T	T	F	F	F
26	T	T	F	F	T
27	T	T	F	T	F
28	T	T	F	T	T
29	T	T	T	F	F
30	T	T	T	F	T
31	T	T	T	T	F
32	T	T	T	T	T

Where T is *True* and F is *False*. The next task is to express the following facts about the *Wumpus world* in logical sentences R_i.

- The wumpus, W, only occurs in an adjacent square to a stench, S.
- There is only one wumpus, W.
- There is no stench, S, perceived in squares $(2,1)$ and $(1,1)$
- There is a perceived stench, S, in squares $(1,1)$

These correspond to the following sentences in the current instance of the world.

$$
\begin{aligned}
&R_1 : (W_{1,3} \Leftrightarrow S_{1,2}) \lor (W_{2,2} \Leftrightarrow S_{1,2} \lor S_{2,1}) \lor (W_{3,1} \Leftrightarrow S_{2,1}) \\
&R_2 : W_{1,3} \lor W_{2,2} \lor W_{3,1} \\
&R_3 : \neg(S_{2,1} \land S_{1,1}) \\
&R_4 : S_{1,2}
\end{aligned}
$$

Below a plot of the *Wumpus world* after the agent has reached square (2,1). This square is reached after first travelling to square (1,2), back to square (1,1) and finally (2,1). The plot also depicts the world before any deductions is made. This is why square (2,2) says both OK and W?. In the plot below OK means that the square is safe, W? means that there is a possibility of a *Wumpus* in the square. S denotes a stench in the considered square. Finally Agent reffers to the player or agent traversing the world.

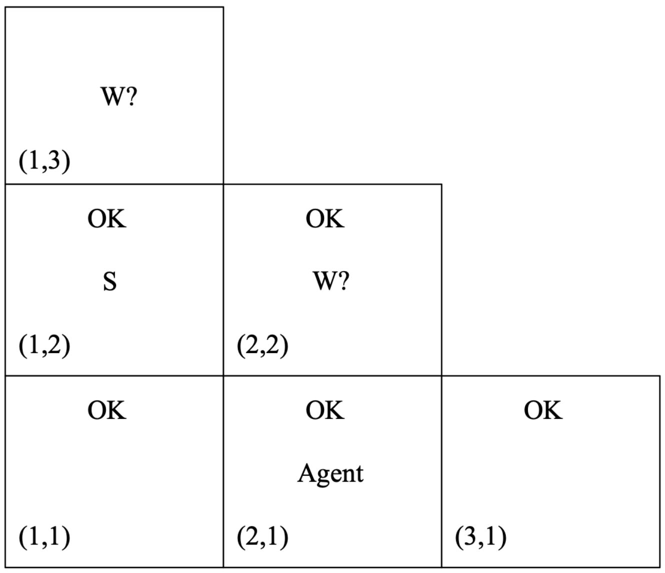

Figure 1: View of the world when agent has reached square (2,1)

Table 2: Table of all possible models with corresponding truth values

Index	$S_{1,2}$	$S_{2,1}$	$W_{1,3}$	$W_{2,2}$	$W_{3,1}$	R_1	R_2	R_3	R_4	Knowledge base
1	F	F	F	F	F	T	F	T	F	F
2	F	F	F	F	T	F	T	T	F	F
3	F	F	F	T	F	F	T	T	F	F
4	F	F	F	T	T	F	F	T	F	F
5	F	F	T	F	F	F	T	T	F	F
6	F	F	T	F	T	F	F	T	F	F
7	F	F	T	T	F	F	F	T	F	F
8	F	F	T	T	T	F	F	T	F	F
9	F	T	F	F	F	F	F	F	F	F
10	F	T	F	F	T	T	T	F	F	F
11	F	T	F	T	F	T	T	F	F	F
12	F	T	F	T	T	T	F	F	F	F
13	F	T	T	F	F	F	T	F	F	F
14	F	T	T	F	T	F	F	F	F	F
15	F	T	T	T	F	F	F	F	F	F
16	F	T	T	T	T	F	F	F	F	F
17	T	F	F	F	F	F	F	T	T	F
18	T	F	F	F	T	F	T	T	T	F
19	T	F	F	T	F	F	T	T	T	F
20	T	F	F	T	T	F	F	T	T	F
21	T	F	T	F	F	T	T	T	T	T
22	T	F	T	F	T	F	F	T	T	F
23	T	F	T	T	F	T	F	T	T	F
24	T	F	T	T	T	F	F	T	T	F
25	T	T	F	F	F	F	F	F	T	F
26	T	T	F	F	T	F	T	F	T	F
27	T	T	F	T	F	T	T	F	T	F
28	T	T	F	T	T	T	F	F	T	F
29	T	T	T	F	F	F	T	F	T	F
30	T	T	T	F	T	T	F	F	T	F
31	T	T	T	T	F	T	F	F	T	F
32	T	T	T	T	T	T	F	F	T	F

Where, again, T is $True$ and F is $False$.

This concludes the knowledge base to containing the following to statements about the world.

- There is stech in square (1,2) and a Wumpus in square (1,3)

This means that two models of the world can be constructed. The model of the knowledge base, $M(KB)$, and the model also containing the fact that there is a Wumpus in square $(1,3)$, $W_{1,3}$, $M(W_{1,3})$. Furthermore this means the following.

$$KB \models W_{1,3}$$

Since.

$$M(KB) \subseteq M(W_{1,3})$$

1.1 b)

The problem is defined as the following. There are three unique doors with the following writing on each door.

- Door 1: The treasure is not here
- Door 2: The treasure is behind Door 1
- Door 3: The treasure is behind Door 2

Furthermore, it is known that the writing on two of the doors are false while the third text is true.

The following notation is introduced. The treasure is defined as T_i, where i is the Door that the sentence consideres. The Doors are defined as D_i where i is the Door that is considered. This means that the problem is a set of logical sentences. These sentences has two versions each, one for when the statement on the Door is true and one when it is false.

$$\begin{aligned}
&R_1 : \neg T_1 \\
&R_2 : T_1 \\
&R_3 : T_2 \\
&R_4 : (R_1 \land \neg R_2 \land \neg R_3) \lor (\neg R_1 \land R_2 \land \neg R_3) \land (\neg R_1 \land \neg R_2 \land R_3)
\end{aligned}$$

From this set of atomic sentences the following truth table can be constructed.

Table 3: Table of all possible models with corresponding truth values

Index	R_1	R_2	R_3	R_4	Result
1	F	F	T	T	Contradiction
2	F	T	F	T	T_1
3	T	F	F	T	T_3

From the tuthe table above the following conclusions can be drawn. Door number 1 can not be the door that has the true insciption since this leads to a contradiction. However any of the other doors can be true, therefore it is not possible to fully determine where the treasure is located. The treasure can thus be either behind door 1 or door 3.

1.2

In the following task a family tree is to be explained using first order logic. To do this some basic predicates need to be defined.

$$Parent(p_1, p_2) = \text{"}p_1 \textit{ is the parent of } p_2\text{"} \tag{1}$$
$$Spouse(p_1, p_2) = \text{"}p_1 \textit{ is the spouse of } p_2\text{"} \tag{2}$$

Using these definitions the rest of the predicates can be defined using first order logic. More precisely, the following predicates can be defined using the definition of *Parent*, (1).

Child:

$$\forall p_1, p_2(Child(p_1, p_2)) \Leftrightarrow Parent(p_2, p_1) \tag{3}$$

Greatgrandparent:

$$\forall p_1, p_2(Greatgrandparent(p_1, p_2)) \Leftrightarrow \exists p_3, p_4(Parent(p_3, p_2) \land Parent(p_4, p_3) \land Parent(p_1, p_4)) \tag{4}$$

Sibling:

$$\forall p_1, p_2(Sibling(p_1, p_2)) \Leftrightarrow \exists p_3(Parent(p_3, p_1) \land Parent(p_3, p_2)) \tag{5}$$

Ancestor:

$$\forall p_1, p_2(Ancestor(p_1, p_2)) \Leftrightarrow Parent(p_1, p_2) \lor (\exists p_3(Parent(p_1, p_3) \land Ancestor(p_3, p_2))) \tag{6}$$

The following definition uses definition (3), *Child*.

Grandchild:

$$\forall p_1, p_2(Grandchild(p_1, p_2)) \Leftrightarrow \exists p_3(Child(p_3, p_2) \land Child(p_1, p_3)) \tag{7}$$

The following definitions uses (1), *Parent*, aswell as (5), *Sibling*.

FirstCousin:

$$\begin{aligned}\forall p_1, p_2(FirstCousin(p_1, p_2)) \Leftrightarrow \exists p_3, p_4(Sibling(p_3, p_4) \land ((Parent(p_3, p_1) \land Parent(p_4, p_2)) \lor \\ (Parent(p_3, p_2) \land Parent(p_4, p_1))))\end{aligned} \tag{8}$$

Pibling:

$$\forall p_1, p_2(Pibling(p_1, p_2)) \Leftrightarrow \exists p_3(Parent(p_3, p_2) \land Sibling(p_1, p_3)) \tag{9}$$

The following definition utilizes (2), *Spouse*, and (5), *Sibling*.

Sibling in law:

$$\begin{aligned}\forall p_1, p_2(SiblinginLaw(p_1, p_2)) \Leftrightarrow \exists p_3((Spouse(p_3, p_2) \land Sibling(p_1, p_3)) \\ \lor(Spouse(p_3, p_1) \land Sibling(p_3, p_2)))\end{aligned} \tag{10}$$

With the above the definitions the basic facts from the following family tree can be stated.

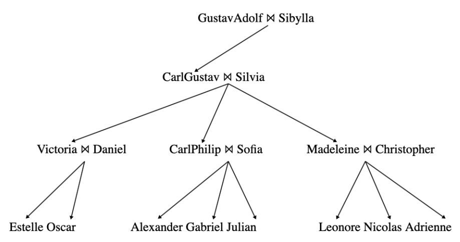

Figure 2: Family tree

$$Spouse(GustavAdolf, Sibylla) \tag{11}$$
$$Parent(GustavAdolf, CarlGustav) \tag{12}$$
$$Parent(Sibylla, CarlGustav) \tag{13}$$
$$Spouse(CarlGustav, Silvia) \tag{14}$$
$$Parent(CarlGustav, Victoria) \tag{15}$$
$$Parent(CarlGustav, CarlPhillip) \tag{16}$$
$$Parent(CarlGustav, Madeleine) \tag{17}$$
$$Parent(Silvia, Victoria) \tag{18}$$
$$Parent(Silvia, CarlPhillip) \tag{19}$$
$$Parent(Silvia, Madeleine) \tag{20}$$
$$Spouse(Victoria, Daniel) \tag{21}$$
$$Spouse(CarlPhillip, Sofia) \tag{22}$$
$$Spouse(Madeleine, Christopher) \tag{23}$$
$$Parent(Victoria, Estelle) \tag{24}$$
$$Parent(Victoria, Oscar) \tag{25}$$
$$Parent(Daniel, Estelle) \tag{26}$$
$$Parent(Daniel, Oscar) \tag{27}$$
$$Parent(CarlPhillip, Alexander) \tag{28}$$
$$Parent(CarlPhillip, Gabriel) \tag{29}$$
$$Parent(CarlPhillip, Julian) \tag{30}$$
$$Parent(Sofia, Alexander) \tag{31}$$
$$Parent(Sofia, Gabriel) \tag{32}$$
$$Parent(Sofia, Julian) \tag{33}$$
$$Parent(Madeleine, Leonore) \tag{34}$$
$$Parent(Madeleine, Nicolas) \tag{35}$$
$$Parent(Madeleine, Adrienne) \tag{36}$$
$$Parent(Christopher, Leonore) \tag{37}$$
$$Parent(Christopher, Nicolas) \tag{38}$$
$$Parent(Christopher, Adrienne) \tag{39}$$

From this list of basic facts the theorems can be derived. To begin with "$Alexander\ is\ CarlGustav's\ GrandChild$" can be derived using the predicate $GrandChild$, (7), and the basic facts (16) along with (28).

$$Grandchild(Alexander, CarlGustav) \Leftrightarrow Parent(CarlPhillip, Alexander) \land Parent(CarlGustav, CarlPhillip)$$

The next theorem, "$Daniel\ and\ CarlPhilip\ are\ sibling-in-law$" can be derived using the predicate $SiblinginLaw$, (10), and basic facts (21), (15) and (16).

$$\begin{gathered}SiblinginLaw(Daniel, CarlPhillip) \Leftrightarrow Spouse(Victoria, Daniel)\land \\ (Parent(CarlGustav, Victoria) \land Parent(CarlGustav, CarlPhillip))\end{gathered}$$

The theorem "$Sibylla\ and\ GustavAdolf\ are\ Gabriel's\ great-grandparents$" is derived by using the predicate $Greatgrandparent$, (4), and the facts (29), (16), (12) and (13).

$$\begin{aligned}&Greatgrandparent(Sibylla, Gabriel) \land Greatgrandparent(GustavAdolf, Gabriel) \Leftrightarrow \\ &Parent(CarlPhillip, Gabriel) \land Parent(CarlGustav, CarlPhillip)\land \\ &Parent(GustavAdolf, CarlGustav) \land Parent(Sibylla, CarlGustav)\end{aligned}$$

The final theorem "$Oscar\ is\ Estelle's\ sibling$" is derived by using the $Sibling$ predicate, (5), and the facts (24) and (25).

$$Sibling(Oscar, Estelle) \Leftrightarrow Parent(Victoria, Estelle) \land Parent(Victoria, Oscar)$$

1.3 a)

This task considers the following bayesian network.

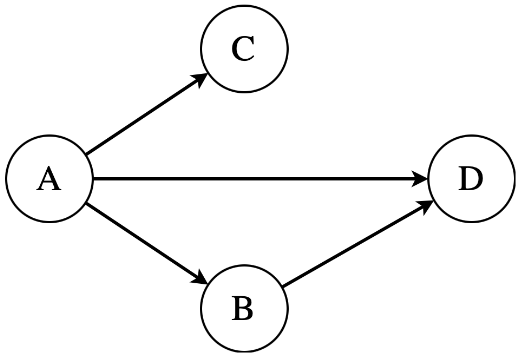

Figure 3: Bayesian network

To begin with the following probability is derived. This derivation follows the standards for a Bayesian network. The probability for the parent node is multiplied by the probability of its child nodes given the parent node. For nodes with multiple parents this probability is defined as the probability of the node itself given all its parents. From this the following formula is derived.

$$P(A, B, C, D) = P(A)P(B|A)P(C|A)P(D|A, B) \tag{40}$$

In order to determine the full probability distribution there is a need to know different parameters. For node A, there is a need to know only 2 parameters. This is because the final parameter can derived from the last 2 since, the probability of any event in A is 1. The same reasoning applies for the rest of the nodes. For nodes B and C there is a need for 2 parameters per given event in the parent. This results in a need for 6 parameters for both node B and node C (2 parameters for the 3 different events in the parent node). For node D, there is a need for 2 parameters for every combination of parent nodes. This results in 18 parameters in total for node D, since the parent nodes have 3 possible events each (9 combinations). This finally results in a need for $2 + 6 + 6 + 18 = 32$ parameters to fully determine the probability distribution. Of course, this does not mean any 32 parameters, since wrong choices of the parameters can result in not determening the full probability distribution. This means that 32 parameters is the minimum ammount of parameters needed.

Then the following probabilities are calculated.

$$i) \quad P(b_1, a_1) = 1 - (P(b_2|a_1) + P(b_3|a_1)) = 0,4 \tag{41}$$

$$ii) \quad P(a_1, b_1, c_1, d_1) = P(a_1)P(b_1|a_1)P(c_1|a_1)P(d_1|a_1, b_1) = 0,01 \tag{42}$$

$$\begin{aligned} iii) \quad P(d_1|a_1) &\equiv \frac{P(d_1, a_1)}{P(a_1)} = \frac{\sum_B P(d_1, a_1, B)}{P(a_1)} = \\ &= \frac{P(a_1)(P(b_1|a_1)P(d_1|a_1, b_1) + P(b_2|a_1)P(d_1|a_1, b_2) + P(b_3|a_1)P(d_1|a_1, b_3)}{P(a_1)} = 0,22 \end{aligned} \tag{43}$$

Since an event in node B is certain to happen for every event in node A the probability $i)$ can be computed. This means that the probability for any event in node B given an event in node A is 1. The second probability simply utelizes the formula derived earlier, (11). For $iii)$ the above formula is used which comes from the book "*Artificial Intelligence A Modern Approach*", page 433 (Russel& Norvig, 2020).

1.3 b)

This task consideres the implementation of python code that checks if a textfile contains adequate information to fully determine the CPT for the Bayesian network in figure 3. For the file to contain adequate information the following criteria must be met.

- The file must contain all conditional probabilities
- All probabilities must lie in the interval [0,1]
- Some probabilites must sum to 1

The first criteria means that the textfile has to have a length of 48. This is because the parent node, A, has 3 events. The child nodes, B and C, each have 3 events. This means that there are 9 possible combinations when the probability of B or C given node A is considered. For the final node, D, there are instead 27 combinations. Since it has 2 different parent nodes. Thus, the textfile must contain the sum of all these combinations, which is 48 conditional probabilities.

The second criteria considers the validity of each probability. A probability is only defined in the domain $[0, 1]$. This is why each value in the texfile must lie in this interval.

The last criteria considers the logic of a Bayesian network. As previously stated, the probability of for example any event in B, b_1, b_2, b_3, given a specific event in node A (its parent node) must sum to 1. This means that if an event happens in node A, something must happen in its child nodes. For node D, this means that for any event in D, given a specific event in both of its parent nodes must sum to 1.

The implementation of this can be seen in full in either the appendix or the seperate file. However, the program is structured in a way that it takes each file that is available and sends it to a function that checks if it contains adequate information. This function is structured in such a way that it goes through each line in the file and counts them. At the same time it extracts the value of each line. If this value is in the correct interval it is counted. Later on the function checks wether or not the length of the file aswell as the ammount of values in the correct interval is correct (48). The final criteria is checked by placing the information from the file in to a dictionary. A vector with the correct order of the conditional probabilities is also constructed. The function then loops through the correct order and sums the values from the dictionary in groups of three. Since the order is garanteed to be correct it is also known that every group of three is supossed to sum to 1, as reasoned earlier. If all of these checks pass the file is sent to a different function that prints a success message. If any check fails the file is instead sent to a function that prints a message that the file has indeed failed.

The above described function can be seen below.

```python
def test_file(filename):
    """Function testing the CPT's of a given file. The conditions are the following.
    The file has to have a length of 48, since this is the number of data points needed to fullfill the CPT (in this case).
    Each value needs to lie in the range [0,1].
    The sum of the probabilities for a child node given the parent node needs to be 1.
    """
    #Initiate variables needed later on
    correct_length = 48

    test_success = 0
    length = 0
    correct_values = 0
    summa = 0

    data = {}

    #Read data file and put data into a dictionary
    with open(filename) as f:
        for row in f:
            length += 1

            (key,val) = row.split("=")

            if float(val) < 1 and float(val) > 0:
                correct_values += 1

            data[key] = float(val)
    f.close()

    #Define correct order
    order = ['a1','a2','a3','b1|a1','b2|a1','b3|a1','b1|a2','b2|a2','b3|a2','b1|a3','b2|a3','b3|a3','c1|a1',\
        'c2|a1','c3|a1','c1|a2','c2|a2','c3|a2','c1|a3','c2|a3','c3|a3','d1|a1,b1','d2|a1,b1','d3|a1,b1','d1|a1,b2',\
        'd2|a1,b2','d3|a1,b2','d1|a1,b3','d2|a1,b3','d3|a1,b3','d1|a2,b1','d2|a2,b1','d3|a2,b1','d1|a2,b2','d2|a2,b2','d3|a2,b2',\
        'd1|a2,b3','d2|a2,b3','d3|a2,b3','d1|a3,b1','d2|a3,b1','d3|a3,b1','d1|a3,b2','d2|a3,b2','d3|a3,b2','d1|a3,b3','d2|a3,b3','d3|a3,b3']

    #Test if file is OK (correct length, all values in [0,1] and every 3 elements should sum to 1)
    if length == correct_length:
        if correct_values == correct_length:
            counter = 0
            for i in order:
                if counter < 3:
                    summa += data[i]
                else:
                    counter = 0
                    if summa == 1:
                        test_success += 1
                        summa = data[i]
                    else:
                        print_fail(filename)
                        return
                counter += 1
            if test_success == 15:
                print_success(filename)
                return
        else:
            print_fail(filename)
            return
    else:
        print_fail(filename)
        return
```

2 References

Russel, S., & Norvig, P. (2020). *Artificial Intelligence A Modern Approach*. Pearson.

3 Appendix

3.1 Python code

```
"""
Python test file, to see if the files are correct CPT's of a Bayesian network or not

Author: Benjamin Elm Jonsson 2023, benjamin.elmjonsson@gmail.com
"""

def test_file(filename):
    """Function testing the CPT's of a given file. The conditions are the following.
    The file has to have a length of 48, since this is the number of data points needed to fullfill the CPT (in this case).
    Each value needs to lie in the range [0,1].
    The sum of the probabilities for a child node given the parent node needs to be 1.
    """
    #Initiate variables needed later on
    correct_length = 48

    test_success = 0
    length = 0
    correct_values = 0
    summa = 0

    data = {}

    #Read data file and put data into a dictionary
    with open(filename) as f:
        for row in f:
            length += 1

            (key,val) = row.split("=")

            if float(val) < 1 and float(val) > 0:
                correct_values += 1

            data[key] = float(val)
    f.close()

    #Define correct order
    order = ['a1','a2','a3','b1|a1','b2|a1','b3|a1','b1|a2','b2|a2','b3|a2','b1|a3','b2|a3','b3|a3','c1|a1',\
        'c2|a1','c3|a1','c1|a2','c2|a2','c3|a2','c1|a3','c2|a3','c3|a3','d1|a1,b1','d2|a1,b1','d3|a1,b1','d1|a1,b2',\
        'd2|a1,b2','d3|a1,b2','d1|a1,b3','d2|a1,b3','d3|a1,b3','d1|a2,b1','d2|a2,b1','d3|a2,b1','d1|a2,b2','d2|a2,b2','d3|a2,b2',\
```

```python
        'd1|a2,b3','d2|a2,b3','d3|a2,b3','d1|a3,b1','d2|a3,b1','d3|a3,
            b1','d1|a3,b2','d2|a3,b2','d3|a3,b2','d1|a3,b3','d2|a3,b3',
            'd3|a3,b3']

    #Test if file is OK (correct length, all values in [0,1] and every
         3 elements should sum to 1)
    if length == correct_length:
        if correct_values == correct_length:
            counter = 0
            for i in order:
                if counter < 3:
                    summa += data[i]
                else:
                    counter = 0
                    if summa == 1:
                        test_success += 1
                        summa = data[i]
                    else:
                        print_fail(filename)
                        return
                counter += 1
            if test_success == 15:
                print_success(filename)
                return
        else:
            print_fail(filename)
            return
    else:
        print_fail(filename)
        return


def print_fail(filename):
    """Function that prints the message that the file has failed the
       tests and therefore
        does not determine the CPT
    """
    filename = filename.split('/Users/benjaminjonsson/Programmering/
       MMS131/1.3b/')[1]
    print(f"{filename} does not fully determine the CPT")


def print_success(filename):
    """Function that prints the message that the file has passed the
       tests and therefore
        does determine the CPT
    """
    filename = filename.split('/Users/benjaminjonsson/Programmering/
       MMS131/1.3b/')[1]
    print(f"{filename} fully determiens the CPT")


```

```
def main():
    """Main file that selects each file and sends it to test function
    """
    files = ['inNO1.txt','inNO2.txt','inNO3.txt','inOK1.txt','inOK2.
       txt']
    for i in files:
        filename = '/Users/benjaminjonsson/Programmering/MMS131/1.3b/'
            + i
        test_file(filename)


if __name__ == '__main__':
    """Determines that the file is runnable, initiates the programme.
    """
    main()
```