Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

**Department of Computer Science and Engineering (Data Science)**

**Subject: Probabilistic Graph Models (DJ19DSC5014)**

**AY: 2022-23**

*Bhuvi Ghosh*
*60009210191*

# Experiment 1

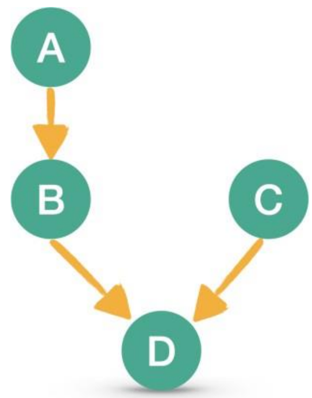
## (Alarm Bayesian Networks)

**Aim:** Solve alarm burglar problem using Bayesian Belief Networks.

**Theory:**

- **Bayesian Belief Networks:** Naive Bayes algorithm is a fast and simple modeling technique used in classification problems. It is used widely due to its speed and relatively good performance, Naive Bayes is built on the assumption that all variables are independent, which in reality is often not true. In some cases, a model may need to be built where there is an option to specify which variables are dependent, independent, or conditionally independent. One may also want to track real-time how event probabilities change as new evidence is introduced to the model. This is where the **Bayesian Belief Networks** come in handy as they allow the construction of a model with nodes and edges by clearly outlining the relationships between variables.
- **Bayesian Belief Networks (BBN) and Directed Acyclic Graphs (DAG):** Bayesian Belief Network (BBN) is a Probabilistic Graphical Model (PGM) that represents a set of variables and their conditional dependencies via a Directed Acyclic Graph (DAG).
To understand what this means, let’s draw a DAG and analyze the relationship between different

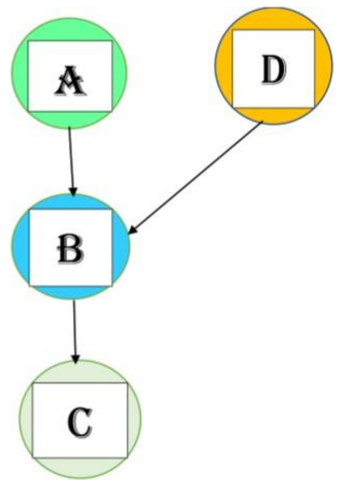
nodes:




- **Independence**: A and C are independent of each other. So are B and C. This is because knowing whether C has happened does not change our knowledge about A or B and vice versa.
- **Dependence**: B is dependent on A since A is the parent of B. This relationship can be written as a conditional probability: P(B|A). D is also dependent on other variables, and in this case, it depends on two of them — B and C. Again, this can be written as a conditional probability: P(D|B,C).
- **Conditional Independence**: D is considered conditionally independent of A. This is because as soon as we know whether event B has happened, A becomes irrelevant from the perspective of D. In other words, the following is true: P(D|B,A) = P(D|B).

- **Discrete Bayesian Networks:** The directed edges (as shown in figure above) of a BN represent conditional distributions. If the values of the vertices are binary, for example, the conditional distributions may be Bernoulli distributions. In case of continuous values, the conditional distributions may be Gaussian. The joint probability distribution is formulated as a product of conditional or marginal probabilities.
  Discrete Baysian Networks are Bayesian Networks which try to determine the probabilities of occurrence of discrete classes similar to a binomial distribution. If the occurrence of only one class is to be predicted then it is said to be a part of a Bernoulli distribution.

- **Bayesian Network Example:**




P(A) denotes the event that you wake up late in the morning.

P(B) denotes the event you are late for school.

P(C) denotes the event you are scolded by your teacher.

P(D) denotes the event there is a traffic jam on the road

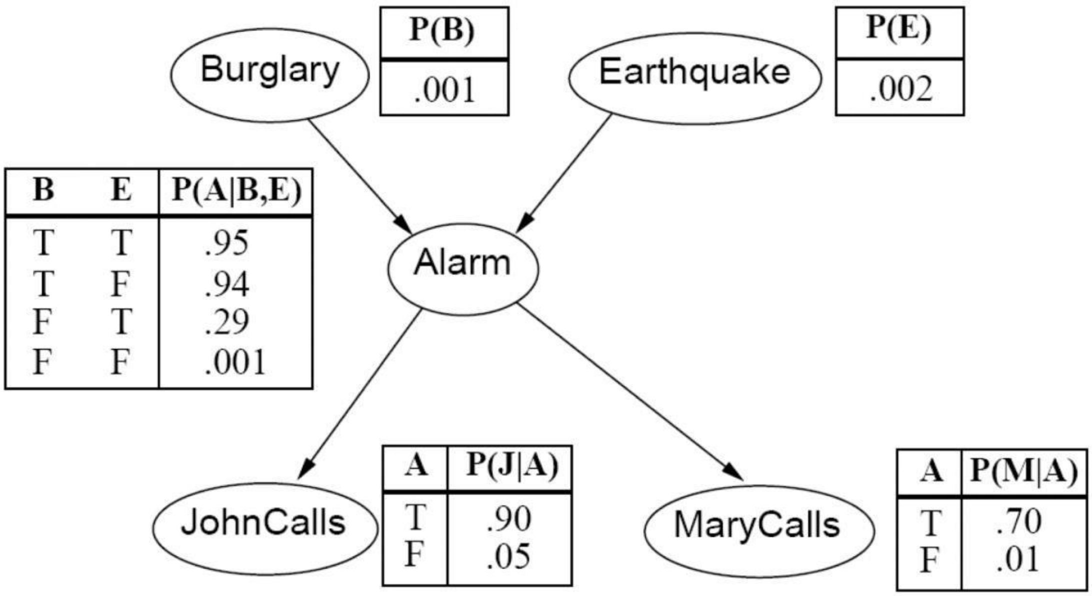
So what is the probability that you are scolded at school? Now, you can easily find that out using the formula. So the final result becomes.

$$P(C|B).P(B|A,D).P(D).P(A)$$

**Burglar Alarm Example**

- Harry has a burglar alarm that is sometimes set off by minor earthquakes. My two neighbors, John and Mary, promised to call me at work if they heard the alarm.
  - Example inference task: suppose Mary calls and John doesn't call. What is the probability of a burglary?
- Random variables in this example are
  - B = a burglary occurs at your house
  - E = an earthquake occurs at your house
  - A = the alarm goes off
  - J = John calls to report the alarm

- M = Mary calls to report the alarm




**Lab Assignments to complete in this session**

- Consider the probabilities as mentioned in the above Figure.
- Plot the Baysian belief network with the help of **networkx** and **pybbn**
- Take User input on the following:

  Does Burglary take place:

  Is there an Earthquake:

  Does the alarm ring:

  Does Mary call:

  Does John call:
- Depending on the User input **calculate the probability of Harry receiving a call** using Bayes theorem as shown above.

Code for visualizing network :

In [1]:
```
!pip install pybbn
```

In [2]:
```
import pandas as pd # for data manipulation
import networkx as nx # for drawing graphs
import matplotlib.pyplot as plt # for drawing graphs

# for creating Bayesian Belief Networks (BBN)
from pybbn.graph.dag import Bbn
from pybbn.graph.edge import Edge, EdgeType
from pybbn.graph.jointree import EvidenceBuilder
from pybbn.graph.node import BbnNode
from pybbn.graph.variable import Variable
from pybbn.pptc.inferencecontroller import InferenceController
```

In [3]:
```
burglary = BbnNode(Variable(0, 'burglary', ['0', '1']), [0.999, 0.001])
earthquake = BbnNode(Variable(1, 'earthquake', ['0', '1']), [0.998, 0.002])
alarm = BbnNode(Variable(2, 'alarm', ['1', '0']), [0.95, 0.05,0.94,0.06,0.29,0.71


john = BbnNode(Variable(3, 'john', ['No', 'Yes']), [0.90,0.10,0.05,0.95])
mary = BbnNode(Variable(4, 'mary', ['No', 'Yes']), [0.70,0.30,0.01,0.99])
```

In [4]:
```
bbn = Bbn() \
    .add_node(burglary) \
    .add_node(earthquake) \
    .add_node(alarm) \
    .add_node(john) \
    .add_node(mary) \
    .add_edge(Edge(burglary, alarm, EdgeType.DIRECTED)) \
    .add_edge(Edge(earthquake, alarm, EdgeType.DIRECTED)) \
    .add_edge(Edge(alarm, john, EdgeType.DIRECTED)) \
    .add_edge(Edge(alarm,mary,EdgeType.DIRECTED)) \
```

```
join_tree = InferenceController.apply(bbn)
```

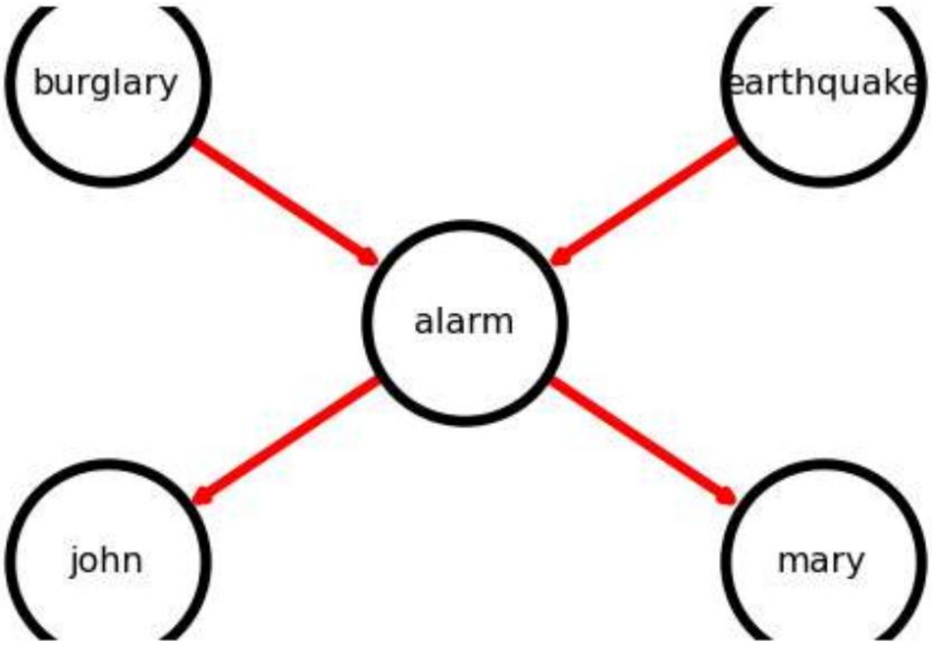
```
# Set node positions
pos = {0: (-1, 2), 1: (1, 2), 2: (0, 0), 3: (-1, -2),4: (1,-2)}

# Set options for graph looks
options = {
    "font_size": 16,
    "node_size": 8000,
    "node_color": "white",
    "edgecolors": "black",
    "edge_color": "red",
    "linewidths": 5,
    "width": 5,}
```

```
# Generate graph
n, d = bbn.to_nx_graph()
nx.draw(n, with_labels=True, labels=d, pos=pos, **options)

# Update margins and print the graph
ax = plt.gca()
ax.margins(0.10)
plt.axis("off")
plt.show()
```

OUTPUT :



Code for finding posterior probability:

```
def print_probs():
    for node in join_tree.get_bbn_nodes():
        potential = join_tree.get_bbn_potential(node)
        print("Node:", node)
        print("Values:")
        print(potential)
        print('----------------')
```

```
def evidence(ev, nod, cat, val):
    ev = EvidenceBuilder() \
    .with_node(join_tree.get_bbn_node_by_name(nod)) \
    .with_evidence(cat, val) \
    .build()
    join_tree.set_observation(ev)
```

```
a = int(input("Did alarm go off(1/0): "))
evidence('ev1', 'alarm', '1', a)
b = int(input("Did earthquake happen(1/0): "))
evidence('ev', 'earthquake', '1', b)
c = int(input("Did burgalary happen(1/0): "))
evidence('ev3', 'burglary', '1', c)
```

```
print_probs()
```

OUTPUT for posterior probabilities:

Current values:

```
Node: 2|alarm|1,0
Values:
2=1|0.94932
2=0|0.05068
----------------
Node: 3|john|No,Yes
Values:
3=No|0.85692
3=Yes|0.14308
----------------
Node: 4|mary|No,Yes
Values:
4=No|0.66503
4=Yes|0.33497
----------------
Node: 1|earthquake|0,1
Values:
1=0|0.99800
1=1|0.00200
----------------
Node: 0|burglary|0,1
Values:
0=0|0.99900
0=1|0.00100
----------------
```

Posterior probabilities after evidence:

```
Did alarm go off(1/0): 1
Did earthquake happen(1/0): 0
Did burgalary happen(1/0): 1
Node: 2|alarm|1,0
Values:
2=1|1.00000
2=0|0.00000
----------------
Node: 3|john|No,Yes
Values:
3=No|0.90000
3=Yes|0.10000
----------------
Node: 4|mary|No,Yes
Values:
4=No|0.70000
4=Yes|0.30000
----------------
Node: 1|earthquake|0,1
Values:
1=0|0.00000
1=1|1.00000
----------------
Node: 0|burglary|0,1
Values:
0=0|0.00000
0=1|1.00000
----------------
```