Laporan Uji Coba Praktikum Probabilistic Learning Kecerdasan Komputasional (KK) - RKA Kelas N

Nama : Abdan Hafidz

NRP : 5054231021

In [6]:
```
from probability import *
from utils import print_table
from notebook import psource, pseudocode, heatmap
```

BAYESIAN NETWORKS

Bayesian Network atau Jaringan Bayes merupakan suatu model probabilistik yang digunakan untuk merepresentasikan distribusi probabilitas bersama (joint probability distribution) dari sejumlah variabel, dengan menyandikan hubungan ketergantungan bersyarat (conditional independence) antarvariabel tersebut. Dalam konteks ini, Jaringan Bayes hanya berfokus pada variabel biner atau boolean, sehingga memungkinkan penyederhanaan analisis probabilitas yang melibatkan kondisi-kondisi yang bernilai benar atau salah.

Struktur Implementasi

Implementasi dari Bayesian Network dilakukan dalam bentuk dua kelas utama, yaitu:

1. **BayesNet** : Merupakan kelas utama yang mewakili keseluruhan jaringan. Kelas ini terdiri atas beberapa node yang masing-masing mewakili satu variabel acak dalam jaringan. Hubungan antarvariabel di dalam jaringan diatur sedemikian rupa untuk menggambarkan ketergantungan bersyarat antara satu variabel dengan variabel lain yang menjadi "orang tua" (parent) dari variabel tersebut.
2. **BayesNode** : Merupakan kelas yang digunakan untuk merepresentasikan setiap node dalam jaringan. Setiap node ini terkait dengan sebuah variabel dan memiliki tabel probabilitas bersyarat atau **conditional probability table (cpt)** . Tabel ini menyimpan nilai probabilitas dari variabel node yang bersangkutan, dengan kondisi bergantung pada variabel orang tua (parent) yang mempengaruhinya, dinyatakan dalam **P(X | parents)** .

BayesNode dan Conditional Probability Table (CPT)

Pada implementasi BayesNode, setiap node memiliki conditional probability table (CPT) yang berfungsi sebagai penyimpan nilai probabilitas dari variabel tersebut bergantung pada kondisi parent-nya. Misalnya, jika sebuah node X memiliki variabel orang tua YY**Y**, maka CPT akan menyimpan nilai P(XIY)P(X | Y)**P****(**X**I**Y**)**. CPT inilah yang menjadi dasar perhitungan probabilitas pada Jaringan Bayes, di mana probabilitas pada setiap node akan dihitung berdasarkan kondisi dari node yang menjadi parent-nya.

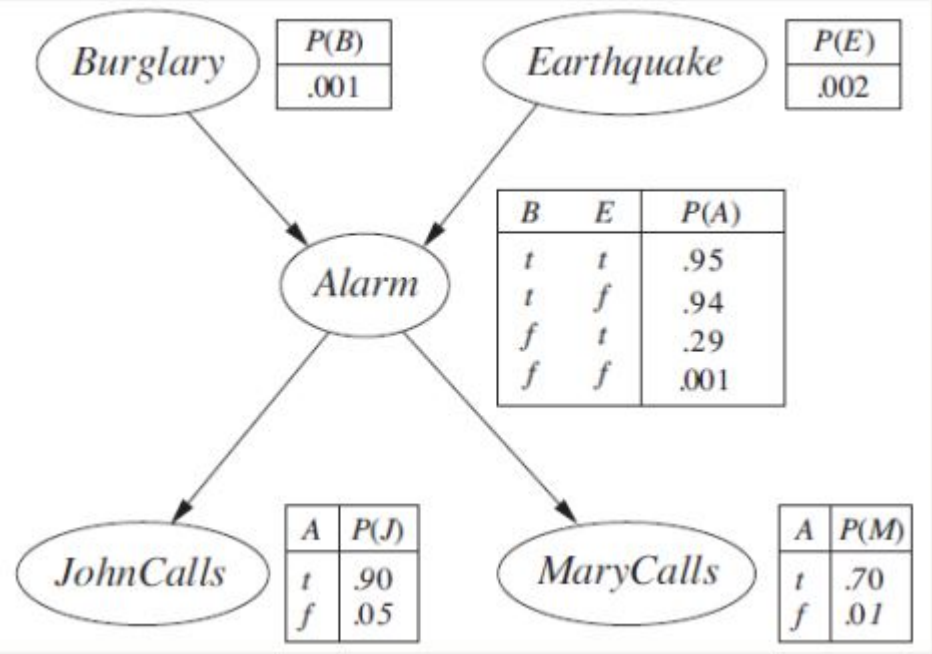
Kesimpulan

Dengan adanya struktur Jaringan Bayes ini, kita dapat menghitung distribusi probabilitas dari suatu variabel yang bergantung pada variabel lain dalam jaringan dengan lebih efisien. Bayesian Network memudahkan perhitungan probabilitas bersyarat yang kompleks dalam data yang mengandung ketergantungan antarvariabel, khususnya dalam masalah yang melibatkan variabel boolean.

In [ ]:
```
psource(BayesNode)
```

Konstruktor menerima tiga parameter: **variable** (nama variabel seperti 'Earthquake'), **parents** (list atau string berisi nama variabel orang tua), dan **cpt** (tabel probabilitas bersyarat). **cpt** adalah dictionary dengan format `{(v1, v2, ...): p, ...}`, di mana `p` merupakan probabilitas dari distribusi `P(X=true | parent1=v1, parent2=v2, ...)`. Kunci dalam dictionary adalah kombinasi nilai boolean yang dimiliki oleh orang tua. Urutan dan jumlah nilai dalam kunci harus sesuai dengan list atau string yang diberikan pada **parents**. Probabilitas `X=false` tidak dicantumkan secara eksplisit, karena dapat dihitung dari `P(X=true)`.

Contoh berikut, diambil dari implementasi jaringan dalam **Gambar 14.3** pada buku, akan memperjelas konsep ini:



Node alarm dapat dibuat sebagai berikut:

In [8]:
```
alarm_node = BayesNode('Alarm', ['Burglary', 'Earthquake'],
                       {(True, True): 0.95,(True, False): 0.94, (False,
True): 0.29, (False, False): 0.001})
```

Kode ini mendefinisikan sebuah *node* Bayesian Network bernama `Alarm` yang bergantung pada dua variabel orang tua, yaitu `Burglary` (pencurian) dan `Earthquake` (gempa bumi). Dengan menggunakan kelas `BayesNode`, kita menetapkan hubungan probabilitas antara kondisi orang tua (`Burglary` dan `Earthquake`) dengan `Alarm`.

Pada *Conditional Probability Table* (CPT) untuk `Alarm`, setiap kombinasi nilai dari orang tua (`True` atau `False`) memiliki probabilitas yang menggambarkan peluang terjadinya `Alarm=True`:

- `(True, True): 0.95` – Jika terjadi pencurian dan gempa, maka probabilitas `Alarm=True` adalah 0.95.
- `(True, False): 0.94` – Jika hanya terjadi pencurian tanpa gempa, probabilitas `Alarm=True` adalah 0.94.
- `(False, True): 0.29` – Jika hanya terjadi gempa tanpa pencurian, probabilitas `Alarm=True` adalah 0.29.
- `(False, False): 0.001` – Jika tidak terjadi pencurian maupun gempa, probabilitas `Alarm=True` sangat kecil, yaitu 0.001.

Dengan kata lain, node `Alarm` ini menangkap hubungan probabilistik yang bersyarat antara pencurian, gempa bumi, dan alarm berbunyi.

Jika hanya ada satu orang tua, penggunaan tuple dapat dihindari. Jadi, format alternatif untuk **cpt** adalah

In [9]:
```
john_node = BayesNode('JohnCalls', ['Alarm'], {True: 0.90, False: 0.05})
mary_node = BayesNode('MaryCalls', 'Alarm', {(True, ): 0.70, (False, ): 0.01}) # Using string for parents.
# Equivalant to john_node definition.
```

Kode ini mendefinisikan dua *node* Bayesian Network yang menggambarkan hubungan probabilitas antara kejadian alarm dan dua orang yang mungkin menelepon, yaitu `John` dan `Mary`.

1. **john_node**:
   - `john_node` adalah *node* yang menunjukkan apakah John akan menelepon berdasarkan kejadian alarm. Node ini bergantung pada variabel orang tua `Alarm`.
   - Conditional Probability Table (CPT) untuk `john_node`:
     - Jika `Alarm=True`, maka probabilitas `JohnCalls=True` adalah 0.90 (probabilitas John menelepon sangat tinggi saat alarm berbunyi).
     - Jika `Alarm=False`, maka probabilitas `JohnCalls=True` adalah 0.05 (probabilitas John menelepon rendah ketika alarm tidak berbunyi).
2. **mary_node**:
   - `mary_node` adalah *node* yang menunjukkan apakah Mary akan menelepon berdasarkan kejadian alarm, dengan cara yang mirip dengan `john_node`, tetapi kali ini menggunakan format string untuk mendefinisikan orang tua.
   - CPT untuk `mary_node`:
     - Jika `Alarm=True`, probabilitas `MaryCalls=True` adalah 0.70.
     - Jika `Alarm=False`, probabilitas `MaryCalls=True` adalah 0.01.

Dengan demikian, kedua *node* ini menggambarkan bagaimana probabilitas seseorang untuk menelepon (John atau Mary) bergantung pada apakah alarm berbunyi atau tidak.

In [10]:
```
burglary_node = BayesNode('Burglary', '', 0.001)
earthquake_node = BayesNode('Earthquake', '', 0.002)
```

Kode ini mendefinisikan dua *node* Bayesian Network, yaitu `Burglary` dan `Earthquake`, yang masing-masing tidak bergantung pada variabel orang tua.

1. **burglary_node**:
    - `burglary_node` adalah *node* yang menunjukkan probabilitas terjadinya pencurian (burglary). Karena tidak ada orang tua yang mempengaruhi, nilai probabilitasnya langsung diberikan.
    - Probabilitas `Burglary=True` adalah 0.001, yang berarti kemungkinan terjadinya pencurian sangat kecil, hanya 0.1%.
2. **earthquake_node**:
    - `earthquake_node` adalah *node* yang menunjukkan probabilitas terjadinya gempa bumi (earthquake). Sama seperti `burglary_node`, ini tidak bergantung pada variabel lain.
    - Probabilitas `Earthquake=True` adalah 0.002, yang berarti kemungkinan terjadinya gempa bumi sedikit lebih kecil dari pencurian, yaitu 0.2%.

Dengan kata lain, kedua *node* ini menggambarkan kejadian-kejadian independen dengan probabilitas yang sangat kecil untuk terjadinya pencurian dan gempa bumi.

Metode **p** pada *node* dapat digunakan untuk mencari probabilitas bersyarat berdasarkan nilai variabel dan kejadian (event). Metode ini menerima dua argumen: **value** dan **event** .

1. **value** : Nilai dari variabel yang kita minati, yaitu `True` atau `False` .
2. **event** : Sebuah dictionary yang berisi pasangan {variabel: nilai}, yang mewakili nilai dari orang tua yang relevan dalam jaringan. Setiap orang tua harus diberi nilai dalam event ini.

Metode **p** akan mengembalikan probabilitas bersyarat **P(X=value | parents=parent_values)** , di mana `parent_values` adalah nilai dari variabel-variabel orang tua yang diberikan dalam event. Dengan kata lain, metode ini menghitung peluang terjadinya suatu variabel (X) dengan kondisi orang tua yang sudah ditentukan.

In [11]:
```
john_node.p(False, {'Alarm': True, 'Burglary': True}) #
P(JohnCalls=False | Alarm=True)
```

Out[11]:
```
0.09999999999999998
```

In [ ]:
```
psource(BayesNet)
```

Konstruktor dari **BayesNet** menerima **node_specs** yang berisi spesifikasi node dan menambahkan setiap **BayesNode** ke dalam objek **nodes** dengan memanggil metode **add** . Metode **add** ini akan menambahkan node ke dalam jaringan (net). Sebelum menambahkan node, orang tua dari node tersebut harus sudah ada dalam jaringan, dan variabel yang bersangkutan belum ada dalam jaringan. Dengan cara ini, metode **add** memungkinkan kita untuk membangun **BayesNet** secara bertahap, dengan memastikan bahwa orang tua node sudah ada sebelum menambahkannya.

Sebagai contoh, **burglary** adalah instance dari **BayesNet** yang sesuai dengan jaringan Bayesian yang dijelaskan di atas.

Berikut adalah definisi **burglary** yang menunjukkan struktur jaringan Bayesian dengan beberapa node:

1. **Burglary** dan**Earthquake** adalah node yang tidak bergantung pada orang tua, masing-masing memiliki probabilitas terjadinya kejadian tersebut (0.001 untuk Burglary dan 0.002 untuk Earthquake).
2. **Alarm** adalah node yang bergantung pada kedua node orang tua,**Burglary** dan**Earthquake** , dengan nilai probabilitas yang tergantung pada kombinasi nilai dari kedua orang tua tersebut.
3. **JohnCalls** dan**MaryCalls** adalah node yang bergantung pada node**Alarm** , dengan probabilitas masing-masing terkait dengan apakah alarm berbunyi atau tidak.

Dengan menggunakan konstruktor **BayesNet** , kita dapat membangun jaringan Bayesian ini dengan urutan penambahan node yang sesuai, memastikan bahwa semua orang tua sudah ada sebelum menambahkan node baru.

```
T, F = True, False

burglary = BayesNet([
    ('Burglary', '', 0.001),
    ('Earthquake', '', 0.002),
    ('Alarm', 'Burglary Earthquake',
     {(T, T): 0.95, (T, F): 0.94, (F, T): 0.29, (F, F): 0.001}),
    ('JohnCalls', 'Alarm', {T: 0.90, F: 0.05}),
    ('MaryCalls', 'Alarm', {T: 0.70, F: 0.01})
])
```

In [13]:
```
burglary
```

Out[13]:
```
BayesNet([('Burglary', ''), ('Earthquake', ''), ('Alarm', 'Burglary Earthquake'),
('JohnCalls', 'Alarm'), ('MaryCalls', 'Alarm')])
```

Metode **variable_node** pada **BayesNet** memungkinkan kita untuk mengakses instance **BayesNode** yang ada di dalam sebuah jaringan Bayesian. Dengan menggunakan metode ini, kita dapat mencapai node tertentu di dalam jaringan dan melakukan modifikasi langsung pada **cpt** (Conditional Probability Table) dari node tersebut.

Ini memungkinkan kita untuk memperbarui atau mengubah probabilitas bersyarat dari suatu variabel dalam jaringan, memberikan fleksibilitas untuk melakukan perubahan pada distribusi probabilitas seiring dengan perubahan kondisi atau kebutuhan analisis.

In [14]:
```
type(burglary.variable_node('Alarm'))
```

Out[14]:
```
probability.BayesNode
```

In [15]:
```
burglary.variable_node('Alarm').cpt
```

Out[15]:
```
{(True, True): 0.95,
 (True, False): 0.94,
 (False, True): 0.29,
 (False, False): 0.001}
```

Kode **burglary.variable_node('Alarm').cpt** digunakan untuk mengakses **cpt** (Conditional Probability Table) dari node **Alarm** dalam jaringan **burglary** .

cpt ini berisi nilai probabilitas bersyarat yang menggambarkan hubungan antara variabel **Alarm** dan orang tuanya, yaitu **Burglary** dan **Earthquake** . Setiap pasangan nilai boolean untuk orang tua (Burglary dan Earthquake) memiliki probabilitas tersendiri untuk **Alarm** menjadi `True` atau `False` .

Berikut adalah isi dari **cpt** tersebut:

- `(True, True): 0.95` – Jika terjadi **Burglary** dan**Earthquake** , maka probabilitas**Alarm** akan `True` adalah 0.95.
- `(True, False): 0.94` – Jika terjadi **Burglary** tetapi tidak terjadi**Earthquake** , maka probabilitas**Alarm** akan `True` adalah 0.94.
- `(False, True): 0.29` – Jika tidak terjadi**Burglary** tetapi terjadi**Earthquake** , maka probabilitas**Alarm** akan `True` adalah 0.29.
- `(False, False): 0.001` – Jika tidak terjadi**Burglary** dan tidak terjadi**Earthquake** , maka probabilitas**Alarm** akan `True` sangat kecil, yaitu 0.001.

Dengan kata lain, kode ini memungkinkan kita untuk melihat dan memodifikasi tabel probabilitas bersyarat untuk node **Alarm** berdasarkan nilai orang tuanya.

In [ ]:
```
psource(enumerate_all)
```

Exact Inference in Bayesian Networks

Jaringan Bayesian adalah cara yang lebih efisien untuk merepresentasikan distribusi gabungan penuh dan memungkinkan kita untuk melakukan inferensi, yaitu menjawab pertanyaan tentang distribusi probabilitas variabel acak dengan mempertimbangkan beberapa bukti yang diberikan.

Namun, algoritma eksak memiliki keterbatasan ketika diterapkan pada jaringan yang lebih besar, karena kinerjanya dapat menurun drastis. Oleh karena itu, algoritma aproksimasi yang lebih efisien untuk jaringan besar akan dibahas pada bagian berikutnya.

Inference by Enumeration

Untuk melakukan inferensi dalam Jaringan Bayesian, kita bisa menggunakan teknik-teknik yang mirip dengan yang digunakan dalam fungsi enumerate_joint_ask dan enumerate_joint. Fungsi enumeration_ask dan enumerate_all mengimplementasikan algoritma yang dijelaskan dalam Gambar 14.9 di buku tersebut, yang membantu dalam proses perhitungan probabilitas berdasarkan informasi yang ada.

enumerate_all secara rekursif menghitung bentuk umum dari **Persamaan 14.4** dalam buku, yaitu:

$$\mathbf{P}(X|\mathbf{e}) = \alpha\mathbf{P}(X,\mathbf{e}) = \alpha\sum_{y}\mathbf{P}(X,\mathbf{e},\mathbf{y})$$

Di sini, **P(X, e, y)** dituliskan sebagai hasil perkalian dari probabilitas bersyarat **P(variable | parents(variable))** yang ada dalam Jaringan Bayesian. Artinya, kita mengalikan probabilitas-probabilitas bersyarat untuk setiap variabel dan orang tuanya dalam jaringan.

Sedangkan **enumeration_ask** memanggil **enumerate_all** untuk setiap nilai variabel yang menjadi query **X** dan kemudian melakukan normalisasi pada hasilnya. Ini memungkinkan kita untuk menghitung probabilitas bersyarat **P(X | e)** dengan menjumlahkan semua kemungkinan kombinasi nilai untuk variabel yang belum diketahui (dalam hal ini **y**) dan kemudian menormalkan hasilnya agar total probabilitas menjadi 1. ```

In [ ]:
```
psource(enumeration_ask)
```

Untuk menyelesaikan masalah **P(Burglary=True | JohnCalls=True, MaryCalls=True)** menggunakan **jaringan Bayesian** (Bayesian Network), kita dapat menggunakan teknik **enumeration_ask** . Proses ini memungkinkan kita untuk menghitung probabilitas bersyarat berdasarkan bukti yang diberikan.

Secara singkat, **enumeration_ask** adalah sebuah metode untuk melakukan inferensi pada jaringan Bayesian, yang berfungsi untuk menghitung nilai probabilitas dari suatu variabel yang kita inginkan (misalnya **Burglary**), berdasarkan bukti yang diberikan (misalnya **JohnCalls** dan **MaryCalls**).

Langkah-langkah dalam **enumeration_ask** :

1. **X** : Ini adalah variabel yang ingin kita temukan probabilitasnya, dalam hal ini**Burglary** .
2. **e** : Ini adalah bukti yang diberikan dalam bentuk dictionary, yaitu nilai-nilai yang kita ketahui. Dalam kasus ini, bukti adalah**JohnCalls=True** dan**MaryCalls=True** .
3. **bn** : Ini adalah jaringan Bayesian yang berisi model probabilitas yang digunakan untuk melakukan inferensi.

Proses perhitungan:

Untuk menghitung **P(Burglary=True | JohnCalls=True, MaryCalls=True)** , kita perlu menghitung probabilitas ter-iterasi dari berbagai variabel dalam jaringan Bayesian, yang melibatkan **Burglary** , **JohnCalls** , dan **MaryCalls** serta variabel-variabel lainnya yang relevan dalam jaringan.

Proses **enumeration_ask** akan mengiterasi semua kemungkinan nilai dari variabel-variabel yang tidak diketahui (misalnya variabel yang tidak terkait langsung dengan bukti) untuk menghitung probabilitas bersyarat. Kemudian, nilai yang dihitung akan disesuaikan dengan bukti yang ada.

Proses ini dapat dijelaskan sebagai berikut:

- Kita menghitung**P(Burglary=True, JohnCalls=True, MaryCalls=True)** .
- Kemudian menghitung**P(JohnCalls=True, MaryCalls=True)** (dengan mengintegrasikan variabel-variabel yang lain).
- Hasil akhirnya adalah pembagian antara dua nilai tersebut untuk mendapatkan probabilitas yang diinginkan.

Dengan demikian, **enumeration_ask** membantu dalam melakukan inferensi yang melibatkan variabel bersyarat dalam jaringan Bayesian, memberikan kita **P(Burglary=True | JohnCalls=True, MaryCalls=True)** sebagai hasil.

Jika Anda memiliki jaringan Bayesian terkait **Burglary** , Anda dapat menggunakan metode ini untuk menghitung probabilitas tersebut dalam implementasi Python.

In [18]:
```
ans_dist = enumeration_ask('Burglary', {'JohnCalls': True, 'MaryCalls': True}, burglary)
ans_dist[True]
```

Out[18]:
```
0.2841718353643929
```

Eliminasi Variabel (Variable Elimination)

Algoritma enumerasi dapat ditingkatkan secara signifikan dengan menghilangkan perhitungan yang berulang. Pada algoritma enumerasi, kita menggabungkan semua variabel tersembunyi (hidden variables) dalam bentuk gabungan (joint). Proses ini menghasilkan ukuran eksponensial berdasarkan jumlah variabel tersembunyi. **Eliminasi variabel** (Variable Elimination) menggunakan teknik yang lebih efisien dengan menggabungkan dua langkah utama, yaitu **penggabungan** (join) dan **marginalisasi** (marginalization), secara bergantian.

Sebelum kita mempelajari implementasi dari **Eliminasi Variabel** , ada baiknya kita memahami terlebih dahulu apa yang dimaksud dengan **Faktor** (Factors).

Apa Itu Faktor?

Secara umum, kita menyebut array multidimensi dari tipe **P(Y1 ... Yn | X1 ... Xm)** sebagai **faktor** , di mana beberapa variabel **Xs** dan **Ys** mungkin memiliki nilai yang sudah ditentukan. Faktor ini dapat dipandang sebagai representasi dari distribusi probabilitas yang tercondition atau bersyarat, dan biasanya digunakan untuk menggambarkan hubungan antara variabel yang terlibat dalam jaringan Bayesian.

Faktor-faktor ini diimplementasikan dalam modul probabilitas menggunakan kelas **Factor** . Kelas **Factor** ini menerima dua input utama:

- **variables** : Variabel-variabel yang terlibat dalam faktor tersebut.

- **cpt** (Conditional Probability Table): Tabel probabilitas bersyarat yang menggambarkan hubungan antara variabel-variabel dalam faktor tersebut.

Fungsi Pembantu (Helper Functions)

Dalam implementasi **Eliminasi Variabel** , ada beberapa **fungsi pembantu** yang digunakan untuk membantu pembuatan **cpt** untuk faktor yang diberikan bukti (evidence). Fungsi-fungsi ini berguna dalam mengkalkulasi distribusi probabilitas dan mempermudah proses marginalisasi serta penggabungan variabel dalam langkah eliminasi.

Beberapa fungsi pembantu yang umum digunakan adalah:

1. **Fungsi untuk marginalisasi** : Membantu menghitung distribusi marginal dari faktor dengan mengeliminasi variabel tertentu.
2. **Fungsi untuk penggabungan** : Menggabungkan dua atau lebih faktor untuk menghasilkan faktor baru yang menggabungkan informasi dari faktor-faktor yang ada.

Proses eliminasi variabel akan mengulangi langkah-langkah penggabungan dan marginalisasi ini untuk mengurangi ukuran faktor dan mempercepat perhitungan probabilitas akhir yang diinginkan.

In [ ]:
```
psource(make_factor)
```

make_factor digunakan untuk membuat **cpt** (Conditional Probability Table) dan **variables** yang akan diteruskan ke konstruktor dari **Factor** . Fungsi **make_factor** digunakan untuk setiap variabel dalam jaringan Bayesian. Fungsi ini menerima tiga argumen utama:

1. **var** : variabel tertentu yang ingin diproses.
2. **e** : bukti yang ingin digunakan untuk melakukan inferensi.
3. **bn** : jaringan Bayesian tempat variabel tersebut berada.

Penjelasan tentang **variables** dan **cpt** :

- **variables** untuk setiap node dalam jaringan Bayesian merujuk pada daftar yang terdiri dari variabel itu sendiri dan orang tua (parents) dari variabel tersebut, dengan pengecualian variabel yang merupakan bagian dari bukti (**evidence**).
  - Langkah pertama adalah mengambil**node.parents** untuk setiap node.
  - Kemudian, kita memfilter variabel orang tua yang ada dalam bukti, dan hanya menyisakan variabel yang tidak ada dalam bukti.
- **cpt** yang dibuat oleh**make_factor** adalah**cpt** yang mirip dengan**cpt** asli dari node, tetapi hanya mencakup baris-baris yang sesuai dengan bukti yang diberikan.
  - Dengan kata lain, hanya nilai-nilai probabilitas yang memenuhi bukti yang dimasukkan dalam**cpt** ini. Ini berarti, jika dalam bukti ada variabel yang diketahui nilainya, maka hanya baris yang mencocokkan nilai tersebut yang akan digunakan dalam**cpt** .

Secara keseluruhan, fungsi **make_factor** memungkinkan kita untuk membuat faktor-faktor yang relevan untuk inferensi probabilitas dalam jaringan Bayesian, dengan

mempertimbangkan bukti yang diberikan dan menyaring informasi yang tidak relevan berdasarkan bukti tersebut.

In [ ]:
```
psource(all_events)
```

Penjelasan Fungsi **all_events** :

- Fungsi**all_events** akan memperhitungkan bukti yang relevan dengan node yang sedang diproses.
- Fungsi ini menghasilkan peristiwa-peristiwa yang konsisten dengan bukti yang ada. Artinya, hanya kombinasi dari variabel-variabel yang sesuai dengan bukti yang akan disertakan dalam keluaran.
- Karena**all_events** adalah generator, ketika dipanggil berulang kali, fungsi ini akan mengembalikan satu peristiwa (event) pada setiap pemanggilan. Proses ini memungkinkan kita untuk mengakses peristiwa-peristiwa yang mendukung bukti secara bertahap dan efisien.

Contoh Penggunaan:

Sebagai contoh, kita dapat mencoba menggunakan fungsi ini dengan referensi pada **Page 524** dari buku yang disebutkan. Misalnya, kita ingin membuat **f** $_5$(A) = P(m | A), yang berarti kita ingin menghitung probabilitas **m** (misalnya suatu peristiwa) diberikan bahwa **A** terjadi, menggunakan jaringan Bayesian.

Dengan menggunakan **all_events** , kita dapat menghasilkan semua peristiwa yang relevan dengan bukti **A** , dan menghitung probabilitas berdasarkan peristiwa-peristiwa yang mendukung bukti tersebut. Fungsi generator ini sangat berguna karena memungkinkan kita untuk memproses setiap peristiwa secara iteratif, menghemat memori, dan mempercepat perhitungan probabilitas dalam jaringan Bayesian.

In [21]:
```
f5 = make_factor('MaryCalls', {'JohnCalls': True, 'MaryCalls': True}, burglary)
```

In [22]:
```
f5
```

Out[22]:
```
<probability.Factor at 0x222d1f36010>
```

In [23]:
```
f5.cpt
```

Out[23]:
```
{(True,): 0.7, (False,): 0.01}
```

Pernyataan `f5 = make_factor('MaryCalls', {'JohnCalls': True, 'MaryCalls': True}, burglary)` digunakan untuk membuat sebuah **factor** yang berisi **Conditional Probability Table (CPT)** untuk variabel **MaryCalls** , dengan bukti (**evidence**) bahwa **JohnCalls** dan **MaryCalls** keduanya bernilai **True** , pada jaringan Bayesian **burglary** .

Penjelasan tentang Proses:

1. **Variabel 'MaryCalls'** : Fungsi **make_factor** pertama-tama akan membuat faktor untuk variabel **MaryCalls** . Ini berarti kita tertarik untuk mengetahui bagaimana probabilitas **MaryCalls** berhubungan dengan variabel-variabel lainnya dalam jaringan Bayesian, khususnya dalam konteks bukti yang diberikan.
2. **Bukti ('evidence')** : Bukti yang diberikan dalam bentuk dictionary adalah `{'JohnCalls': True, 'MaryCalls': True}` . Ini berarti kita sudah mengetahui bahwa kedua variabel **JohnCalls** dan **MaryCalls** memiliki nilai **True** . Dalam konteks jaringan Bayesian, bukti ini akan digunakan untuk menyaring atau membatasi **CPT** yang relevan, hanya menyertakan kemungkinan-kemungkinan yang sesuai dengan bukti ini.
3. **Jaringan Bayesian ('burglary')** : **burglary** adalah nama jaringan Bayesian yang digunakan, yang mencakup variabel-variabel seperti **Burglary** , **JohnCalls** , dan **MaryCalls** . Dalam jaringan ini, **JohnCalls** dan **MaryCalls** biasanya tergantung pada **Burglary** , dengan **Burglary** sebagai faktor yang mempengaruhi apakah **JohnCalls** dan **MaryCalls** akan bernilai **True** atau tidak.

Apa yang Terjadi di Dalam Fungsi **make_factor** :

- Fungsi **make_factor** akan membuat **variables** yang relevan untuk faktor tersebut. Dalam hal ini, **variables** akan mencakup **MaryCalls** dan orang tua (parents) dari **MaryCalls** dalam jaringan **burglary** . Kemudian, setiap variabel yang ada dalam bukti (yaitu **JohnCalls** dan **MaryCalls**) akan dihapus dari daftar variabel yang akan diproses.
- **CPT (Conditional Probability Table)** : Fungsi ini juga akan mengurangi **CPT** yang ada pada node **MaryCalls** hanya dengan baris-baris yang sesuai dengan bukti yang diberikan. Dalam hal ini, hanya baris yang mencakup **JohnCalls = True** dan **MaryCalls = True** yang akan dipertimbangkan dalam **CPT** .

Hasilnya:

Setelah menjalankan pernyataan ini, hasilnya adalah sebuah **factor** yang mencakup distribusi probabilitas untuk **MaryCalls** , mengingat bahwa **JohnCalls** sudah pasti **True** dan **MaryCalls** juga sudah pasti **True** . Faktor ini bisa digunakan untuk menghitung probabilitas atau untuk menggabungkan dengan faktor lain dalam proses inferensi jaringan Bayesian.

Singkatnya, **f5** adalah sebuah **factor** yang berisi informasi probabilitas terkait dengan **MaryCalls** setelah mempertimbangkan bukti bahwa **JohnCalls** dan **MaryCalls** keduanya bernilai **True** dalam jaringan **burglary** .