Introduction to Artificial Intelligence

Roman Barták
Department of Theoretical Computer Science and Mathematical Logic

Agents may need to handle **uncertainty** due to:

- **partial observability** (agent may not know for certain the state, where it is)
- **nondeterminism** (agent may not know where it will end up after performing its action)

Logical agent can:

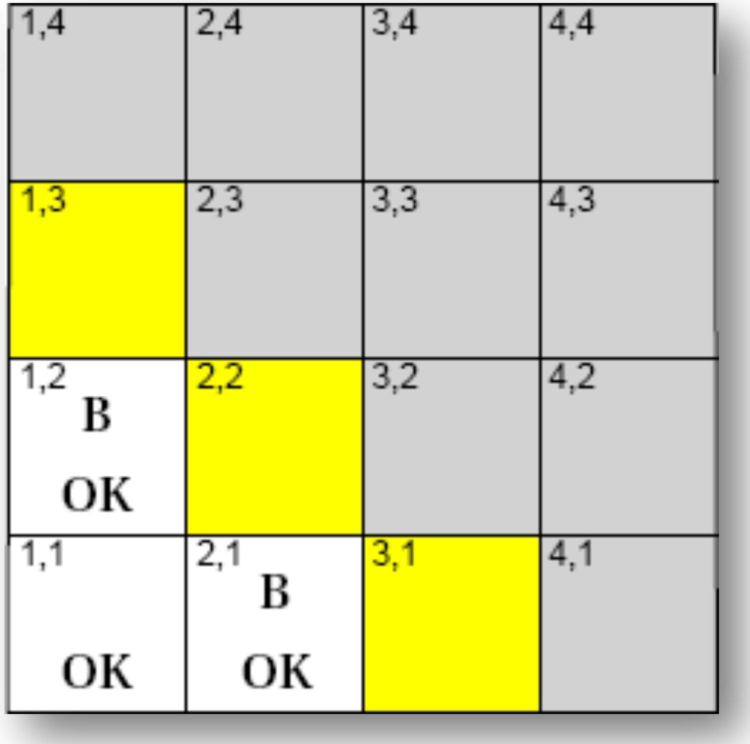
- work with **belief states** (belief state = a set of possible world states, where it might be in)
- generate **contingency plans** (contingency plan handles every possible eventuality).

But, belief-state representations and contingency plans can be impossible **large** and **complex** as they need to cover every possible explanation of observation and every eventuality.

A logical agent believes each sentence to be true or false or has no opinion.

A **probabilistic agent** may have a numerical degree of belief between 0 (certainly false) and 1 (certainly true).

We have a maze with pits that are detected in neighboring squares via breeze (Wumpus and gold will not be assumed now).



Where does the agent should go, if there is breeze at (1,2) and (2,1)?

Each cell – (1,3), (2,2), (3,1) – might contain a pit. Pure logical inference can conclude nothing about which square is most likely to be safe!

=> a logical agent might have to choose randomly

Can we do better?

Like logical assertions, probabilistic assertions are about possible worlds – **sample space** Ω.

- the possible worlds are **mutually exclusive** and **exhaustive**

Each **possible world** ω is associated with a numerical probability $P(\omega)$ such that:

$0 \leq P(\omega) \leq 1$, $\Sigma_{\omega \in \Omega} P(\omega) = 1$

Example: If we are about to roll two (distinguishable) dice, there are 36 possible worlds to consider: (1,1), (1,2),…, (6,6), $P(\omega) = 1/36$

The sets of possible worlds are called **events**.

The probability of event is the sum of probabilities of possible worlds in the event.

$P(\phi) = \Sigma_{\omega \in \phi} P(\omega)$

These probabilities are called **unconditional** or **prior probabilities** („priors" for short).

Example: *P(doubles) = 1/36+1/36+1/36+1/36+1/36+1/36 = 1/6*

Frequently, we have some information, called **evidence** (**b**), and we are interested in probability of some event (**a**).

$P(a \mid b) = P(a \wedge b) / P(b)$, whenever $P(b) > 0$

This is called **conditional** or **posterior probability**

Example: What is the probability of double if we already know that first die rolled to 5?

$P(doubles \mid Die_1 = 5) = 1/36 / (6*1/36) = 1/6$

This can be also written in a different form called the **product rule:**

$P(a \wedge b) = P(a \mid b) . P(b)$

We can refer to a possible world using a **factored representation** – a possible world is represented by a set of variable/value pairs.

Variables in probability theory are called **random variables**. Every random variable has a **domain** – the set of possible values it can take on (similarly to a CSP).

Die_1 – represents a value on the first die 1 (1,…,6)

Cavity – describes whether the patient has or has not cavity (true, false)

A possible world is fully identified by values of all random variables.

$P(Die_1 = 5, Die_2 = 5)$

Probability of all possible worlds can be described using a table called a **full joint probability distribution** – elements are indexed by values of random variables.

	toothache		*¬ toothache*	
	catch	*¬ catch*	*catch*	*¬ catch*
cavity	.108	.012	.072	.008
¬ cavity	.016	.064	.144	.576

To compute posterior probability of a query proposition given observed evidence, we add up probabilities of possible worlds in which the proposition is true (**marginalization** or **summing out**).

$P(\phi) = \Sigma_{\omega:\omega|=\phi} P(\omega)$, $\mathbf{P}(\mathbf{Y}) = \Sigma_{z\in \mathbf{Z}} \mathbf{P}(\mathbf{Y},z)$

A variant of this rule involves conditional probabilities (**conditioning**):

$\mathbf{P}(\mathbf{Y}) = \Sigma_{z\in \mathbf{Z}} \mathbf{P}(\mathbf{Y}|z)P(z)$

In most cases, we are interested in **computing conditional probabilities** of some variables, given **evidence** about others:

$\mathbf{P}(\mathbf{Y} \mid \mathbf{E}=e) = \mathbf{P}(\mathbf{Y}, \mathbf{E}=e) / \mathbf{P}(\mathbf{E}=e)$

We also know $\Sigma_{y\in \mathbf{Y}} \mathbf{P}(\mathbf{Y}=y \mid \mathbf{E}=e) = 1$, so we do not need to calculate $\mathbf{P}(\mathbf{E}=e)$ at all and we can do **normalization** instead:

$\mathbf{P}(\mathbf{Y} \mid \mathbf{E}=e) = \alpha\, \mathbf{P}(\mathbf{Y}, \mathbf{E}=e)$

where $\alpha = 1/ \mathbf{P}(\mathbf{E}=e) = 1/ \Sigma_{y\in \mathbf{Y}} \mathbf{P}(\mathbf{Y}=y, \mathbf{E}=e)$ (normalization constant).

Probabilistic query answering:

In a typical case, we know values **e** of random variables **E** from the **observation** and we are looking for probability distribution of random variables **Y** from the **query**. The other random variables are **hidden H** = **X** – **Y** – **E**.

$\mathbf{P}(\mathbf{Y} \mid \mathbf{E}=e) = \alpha\, \mathbf{P}(\mathbf{Y}, \mathbf{E}=e) = \alpha\, \Sigma_h \mathbf{P}(\mathbf{Y},\mathbf{E}=e,\mathbf{H}=h)$

	toothache		*¬ toothache*	
	catch	*¬ catch*	*catch*	*¬ catch*
cavity	**.108**	**.012**	**.072**	**.008**
¬ cavity	**.016**	**.064**	**.144**	**.576**

P(toothache) (= P(Toothache=true))

$= 0.108 + 0.012 + 0.016 + 0.064 = 0.2$

P(cavity $\vee$ toothache)

$= 0.108+0.012+0.072+0.008+0.016+0.064 = 0.28$

P(cavity|toothache)

$= P(cavity \wedge toothache) / P(toothache)$

$= (0.108 + 0.012) / (0.108 + 0.012 + 0.016 + 0.064) = 0.6$

P($\neg$cavity|toothache)

$= P(\neg cavity \wedge toothache) / P(toothache)$

$= (0.016 + 0.064) / (0.108 + 0.012 + 0.016 + 0.064) = 0.4$

P(Cavity|toothache) = α **P**(Cavity,toothache)

$= \alpha$ [**P**(Cavity,toothache,catch) + **P**(Cavity,toothache,$\neg$catch)]

$= \alpha \, [\langle 0.108, 0.016\rangle + \langle 0.012, 0.064\rangle]$

$= \alpha \, [\langle 0.12, 0.08\rangle] = [\langle 0.6, 0.4\rangle]$

Can we **represent full joint probability distribution more compactly**?

We can exploit **(absolute) independence** of random variables:

$\mathbf{P}(X|Y) = \mathbf{P}(X)$ or $\mathbf{P}(Y|X) = \mathbf{P}(Y)$ or $\mathbf{P}(X,Y) = \mathbf{P}(X).\mathbf{P}(Y)$

Example: two dice $P(Die_1 = 5, Die_2 = 3) = P(Die_1 = 5).P(Die_2 = 3)$

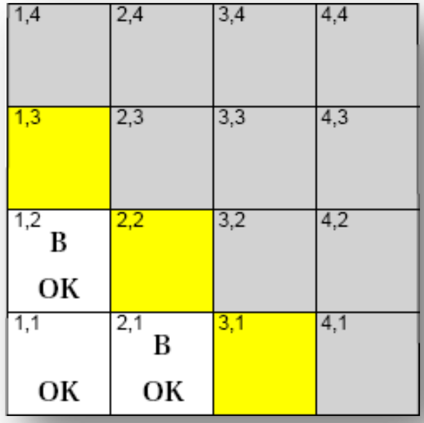
More frequently, two variables X and Y are **conditionally independent** given a third variable Z:

$\mathbf{P}(X|Y,Z) = \mathbf{P}(X|Z)$ or $\mathbf{P}(Y|X,Z) = \mathbf{P}(Y|Z)$ or
$\mathbf{P}(X,Y|Z) = \mathbf{P}(X|Z)\ \mathbf{P}(Y|Z)$

Example:
$\mathbf{P}(Catch, Toothache \mid Cavity) = \mathbf{P}(Catch \mid Cavity).\ \mathbf{P}(Toothache \mid Cavity)$

Toothache and Catch are independent given information about cavity.

One big table can be represented using several smaller tables.



Random Boolean variables:

$P_{i,j}$ – pit at square (i,j)
$B_{i,j}$ – breeze at square (i,j)
(only for the observed squares $B_{1,1}$, $B_{1,2}$ a $B_{2,1}$).

Query to be answered**:**

$\mathbf{P}(P_{1,3} \mid known, b)$.

where we have **evidence**:

$b = \neg b_{1,1} \wedge b_{1,2} \wedge b_{2,1}$

$known = \neg p_{1,1} \wedge \neg p_{1,2} \wedge \neg p_{2,1}$

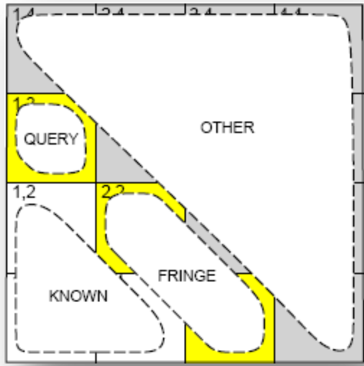
Answer can be computed by enumeration of the full joint probability distribution:

$\mathbf{P}(P_{1,3} \mid known, b) = \alpha \sum_{unknown} \mathbf{P}(P_{1,3}, unknown, known, b)$

where unknown be the variables $P_{i,j}$ except $P_{1,3}$ and known.

However there are $2^{12} = 4096$ terms! Can we do it better (faster)?

unknown = fringe $\cup$ other
from conditional independence we get:
$P(b \mid P_{1,3}, \text{known}, \text{unknown}) = P(b \mid P_{1,3}, \text{known}, \text{fringe})$



$\mathbf{P}(P_{1,3} \mid \text{known}, b)$

$= \alpha\, \Sigma_{\text{unknown}}\, \mathbf{P}(P_{1,3}, \text{known}, \text{unknown}, b)$

product rule $\mathbf{P}(X,Y) = \mathbf{P}(X|Y)\,\mathbf{P}(Y)$

$= \alpha\, \Sigma_{\text{unknown}}\, \mathbf{P}(b \mid P_{1,3}, \text{known}, \text{unknown}) * \mathbf{P}(P_{1,3}, \text{known}, \text{unknown})$

$= \alpha\, \Sigma_{\text{fringe}} \Sigma_{\text{other}}\, \mathbf{P}(b \mid P_{1,3}, \text{known}, \text{fringe},\text{other}) * \mathbf{P}(P_{1,3}, \text{known}, \text{fringe},\text{other})$

$= \alpha\, \Sigma_{\text{fringe}} \Sigma_{\text{other}}\, \mathbf{P}(b \mid P_{1,3}, \text{known}, \text{fringe}) * \mathbf{P}(P_{1,3}, \text{known}, \text{fringe},\text{other})$

$= \alpha\, \Sigma_{\text{fringe}}\, \mathbf{P}(b \mid P_{1,3},\text{known}, \text{fringe}) * \Sigma_{\text{other}}\, \mathbf{P}(P_{1,3}, \text{known},\text{fringe},\text{other})$

$= \alpha\, \Sigma_{\text{fringe}}\, \mathbf{P}(b \mid P_{1,3},\text{known}, \text{fringe}) * \Sigma_{\text{other}}\, \mathbf{P}(P_{1,3}) P(\text{known}) P(\text{fringe}) P(\text{other})$

$= \alpha\, P(\text{known}) \mathbf{P}(P_{1,3})\, \Sigma_{\text{fringe}}\, \mathbf{P}(b \mid P_{1,3},\text{known}, \text{fringe})\, P(\text{fringe})\, \Sigma_{\text{other}}\, P(\text{other})$

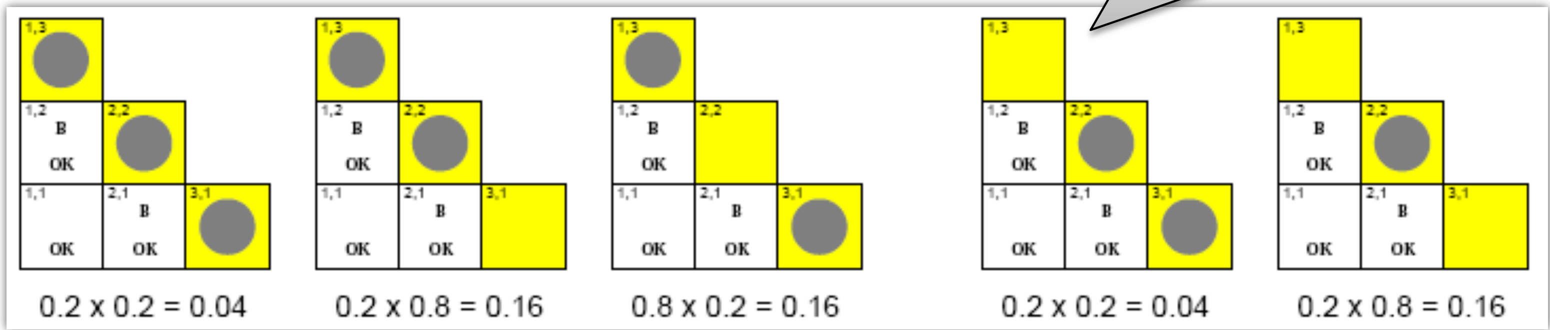
$= \alpha'\, \mathbf{P}(P_{1,3})\, \Sigma_{\text{fringe}}\, \mathbf{P}(b \mid P_{1,3},\text{known}, \text{fringe})\, P(\text{fringe})$

$\alpha' = \alpha.\, P(\text{known})$
$\Sigma_{\text{other}}\, P(\text{other}) = 1$

$\mathbf{P}(P_{1,3} \mid known, b) = \alpha' \, \mathbf{P}(P_{1,3}) \, \Sigma_{fringe} \, \mathbf{P}(b \mid P_{1,3}, known, fringe) \, P(fringe)$

Let us explore possible models (values) of **fringe** that are compatible with observation b.

Let probability of pit be 0.2



$\mathbf{P}(P_{1,3} \mid known, b)$

$= \alpha' \langle 0.2\,(0.04 + 0.16 + 0.16), 0.8\,(0.04 + 0.16) \rangle$

$= \langle 0.31, 0.69 \rangle$

$\mathbf{P}(P_{2,2} \mid known, b) = \langle 0.86, 0.14 \rangle$

Definitely avoid square (2,2)!

Recall the product rule

$P(a \wedge b) = P(a|b)\ P(b) = P(b|a)\ P(a)$

We can deduce a so called **Bayes‘ rule** (law or theorem):

$P(a|b) = P(b|a)\ P(a) / P(b)$

in general:

$$\mathbf{P}(Y|X) = \mathbf{P}(X|Y)\ \mathbf{P}(Y) / \mathbf{P}(X) = \alpha\ \mathbf{P}(X|Y)\ \mathbf{P}(Y)$$

It looks like two steps backward as now we need to know **P**(X|Y), **P**(Y), **P**(X).

But these are the values that we frequently have.

P(cause|effect) = P(effect|cause) P(cause) / P(effect)

- P(effect|cause) describes the **causal direction**
- P(cause|effect) describes the **diagnostic direction**

If all the effects are conditionally independent given the cause variable, we get:

$$\mathbf{P}(Cause, Effect_1, \ldots, Effect_n) = \mathbf{P}(Cause)\ \Pi_i\ \mathbf{P}(Effect_i|Cause)$$

Such a probability distribution is called a **naive Bayes model** (it is often used even in cases where the “effect” variables are not actually conditionally independent given the value of the cause variable).

Medical diagnosis

- from past cases we know P(symptoms|disease), P(disease), P(symptoms)
- for a new patient we know symptoms and looking for diagnosis P(disease|symptoms)

Example:

- meningitis causes a stiff neck 70% of the time
- the prior probability of meningitis is 1/50 000
- the prior probability of stiff neck is 1%

What is the probability that a patient having a stiff neck has meningitis?

$P(m|s) = P(s|m).P(m) / P(s) = 0.7 * 1/50000 / 0.01 = 0.0014$

Why the conditional probability for the diagnostic direction is not stored directly?

- diagnostic knowledge is often more fragile than causal knowledge
- for example, if there is a sudden epidemic of meningitis, the unconditional probability of meningitis P(m) will go up so P(m|s) should also go up while the causal relation P(s|m) is unaffected by the epidemic, as it reflects how meningitis works

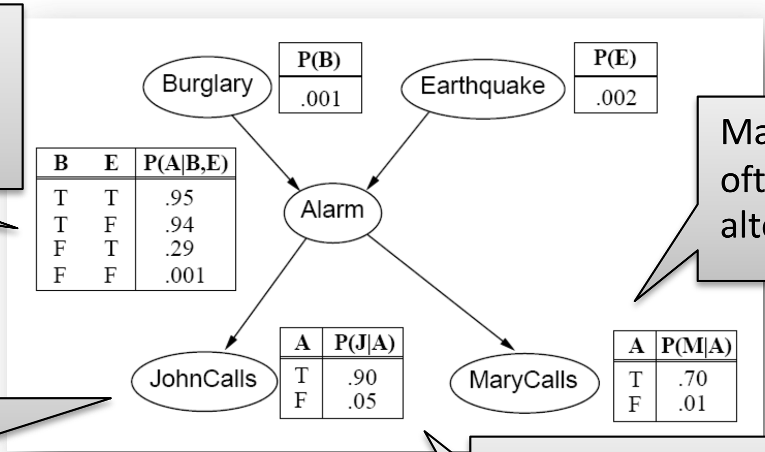
How to represent efficiently any full joint probability distribution by exploiting conditional independence?

Bayesian network specifies **conditional independence relationships** among random variables using a directed acyclic graph (DAG)

- nodes correspond to random variables
- predecessors of nodes are called parents
- each node X has a **conditional probability distribution P**(X | Parents(X))

Burglar alarm is fairly reliable at detecting a burglary, but occasionally responds to minor earthquakes.

John nearly always calls when he hears alarm, but sometimes confuses the telephone ringing with the alarm.



Mary likes loud music and often misses the alarm altogether.

- neighbors do not perceive burglary directly and they do not notice minor earthquakes
- neighbors do not confer (they are independent)

The Bayesian network represents the full joint probability distribution.

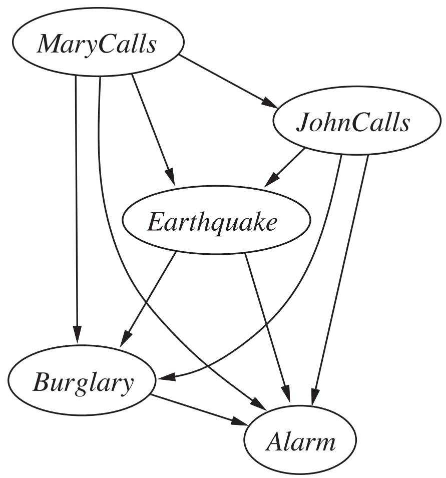
$$P(x_1,\dots,x_n) = \prod_i P(x_i \mid parents(X_i))$$

Nodes:
determine the set of random variables that are required to model the domain and order them

- any order will work, but the resulting networks will be different
- a recommended order is such that causes precede effects (leads to smaller networks and easier-to-fill CPTs)

Arcs:
choose variables X_i in a given order from 1 to n

- in the set $\{X_1,\ldots,X_{i-1}\}$ choose a minimal set of parents for X_i, such that $\mathbf{P}(X_i \mid \text{Parents}(X_i)) = \mathbf{P}(X_i \mid X_{i-1},\ldots,X_1)$ holds
- for each parent insert a link from the parent to X_i
- write down the conditional probability table $\mathbf{P}(X_i \mid \text{Parents}(X_i))$



Why does it work?

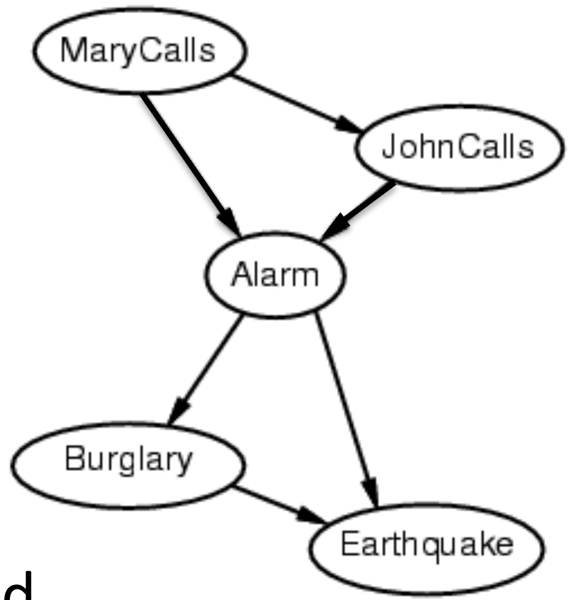
$P(x_1,\ldots,x_n) = \prod_i P(x_i \mid x_{i-1},\ldots,x_1)$ (**chain rule**)

$\mathbf{P}(X_i \mid X_{i-1},\ldots,X_1) = \mathbf{P}(X_i \mid \text{Parents}(X_i))$

where $\text{Parents}(X_i) \subseteq \{X_{i-1},\ldots,X_1\}$

Let us use the following order of random variables:
MarryCalls, JohnCalls, Alarm, Burglary, Earthquake

- MarryCalls has no parents
- if Marry calls then the alarm is probably active which would make it more likely that John calls
- alarm is probably active if Marry or John calls
- if we know the alarm state then the calls from Marry and John do not influence whether the burglary happened



  P(Burglary | Alarm, JohnCalls, MarryCalls) = P(Burglary | Alarm)

- the alarm is an earthquake detector of sorts, but if there was a burglary then it explains the alarm and the probability of an earthquake is only slightly above normal

We introduced Bayesian networks to **do inference** – to deduce posterior probability of some variable(s) **X** from the query given the values **e** of observed variables (evidence), while having the other variables **Y** hidden.

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

where **P**(X,**e**,**y**) is computed as follows

$$P(x_1,\ldots,x_n) = \Pi_i\ P(x_i \mid parents(X_i))$$

```
function ENUMERATION-ASK(X, e, bn) returns a distribution over X
  inputs: X, the query variable
          e, observed values for variables E
          bn, a Bayes net with variables {X} ∪ E ∪ Y   /* Y = hidden variables */

  Q(X) ← a distribution over X, initially empty
  for each value x_i of X do
      Q(x_i) ← ENUMERATE-ALL(bn.VARS, e_{x_i})
          where e_{x_i} is e extended with X = x_i
  return NORMALIZE(Q(X))

function ENUMERATE-ALL(vars, e) returns a real number
  if EMPTY?(vars) then return 1.0
  Y ← FIRST(vars)
  if Y has value y in e
      then return P(y | parents(Y)) × ENUMERATE-ALL(REST(vars), e)
      else return Σ_y P(y | parents(Y)) × ENUMERATE-ALL(REST(vars), e_y)
          where e_y is e extended with Y = y
```

Example:

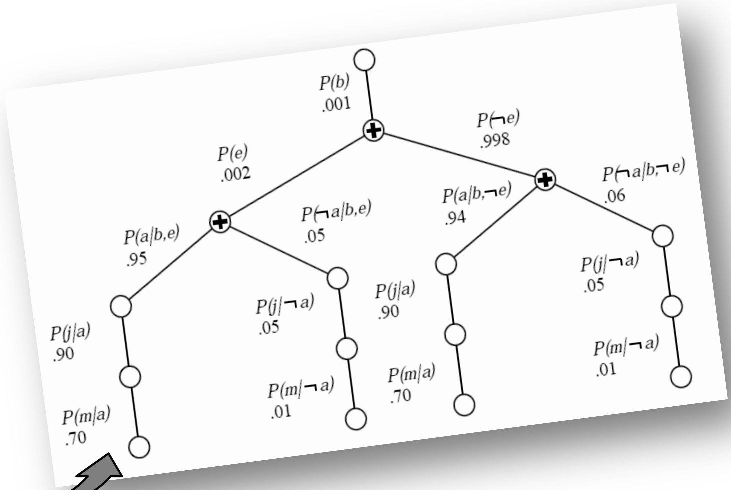
Assume a query about the probability of burglary when both Marry and John calls

$$P(b \mid j,m)$$
$$= \alpha\ \Sigma_e\ \Sigma_a\ P(b)\ P(e)\ P(a|b,e)\ P(j|a)\ P(m|a)$$
$$= \alpha\ P(b)\ \Sigma_e\ P(e)\ \Sigma_a\ P(a|b,e)\ P(j|a)\ P(m|a)$$

The structure of computation can be described using a tree structure.

- it is very similar to solving CSPs and SAT

Notice that some parts are repeated!



Enumeration repeats the same parts of the computation.

We can remember the result and reuse it later (**dynamic programming**).

$\mathbf{P}(B \mid j,m)$

$= \alpha\, \mathbf{P}(B)\, \Sigma_e\, P(e)\, \Sigma_a\, \mathbf{P}(a|B,e)P(j|a)P(m|a)$

$= \alpha\, \mathbf{f_1}(B)\, \Sigma_e\, \mathbf{f_2}(E)\, \Sigma_a\, \mathbf{f_3}(A,B,E)\, \mathbf{f_4}(A)\, \mathbf{f_5}(A)$

Factors $\mathbf{f_i}$ are matrices (tables) corresponding to CPTs.

Evaluation will be done **from right to left**.

- **the product of factors** corresponds to the pointwise product (it is not a multiplication of matrices)
- **summing out a variable** is done by adding up the sub-matrices formed by fixing the variable to each of its values in turn

Notes:

- The algorithm works for any ordering of variables.
- The complexity is given by the size of the largest factor constructed during the operation of the algorithm.
- Eliminate whichever variable minimizes the size of the next factor to be constructed (heuristic).

```
function ELIMINATION-ASK(X, e, bn) returns a distribution over X
  inputs: X, the query variable
          e, observed values for variables E
          bn, a Bayesian network specifying joint distribution P(X1, ..., Xn)

  factors ← []
  for each var in ORDER(bn.VARS) do
      factors ← [MAKE-FACTOR(var, e)|factors]
      if var is a hidden variable then factors ← SUM-OUT(var, factors)
  return NORMALIZE(POINTWISE-PRODUCT(factors))
```

The pointwise **product** of two factors yields a new factor whose variables are the union of the variables from the original factors.

$f(X_1,\ldots,X_j,Y_1,\ldots,Y_k,Z_1,\ldots Z_l) = f(X_1,\ldots,X_j,Y_1,\ldots,Y_k) \,.\, f(Y_1,\ldots,Y_k,Z_1,\ldots Z_l)$

A	B	$f_1(A,B)$
T	T	0.3
T	F	0.7
F	T	0.9
F	F	0.1

*

B	C	$f_2(B,C)$
T	T	0.2
T	F	0.8
F	T	0.6
F	F	0.4

=

A	B	C	$f_3(A,B,C)$
T	T	T	0.06 = 0.3*0.2
T	T	F	0.24 = 0.3*0.8
T	F	T	0.42 = 0.7*0.6
T	F	F	0.28 = 0.7*0.4
F	T	T	0.18 = 0.9*0.2
F	T	F	0.72 = 0.9*0.8
F	F	T	0.06 = 0.1*0.6
F	F	F	0.04 = 0.1*0.4

Then we **sum out** a variable to eliminate it: $\Sigma_a\, f(A,B,C) = f(B,C)$

A	B	C	$f_3(A=T,B,C)$
T	T	T	0.06
T	T	F	0.24
T	F	T	0.42
T	F	F	0.28

+

A	B	C	$f_3(A=F,B,C)$
F	T	T	0.18
F	T	F	0.72
F	F	T	0.06
F	F	F	0.04

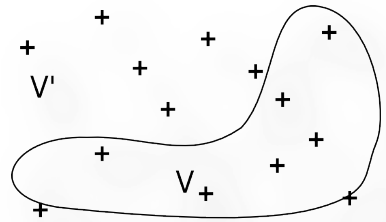
=

B	C	$f_4(B,C)$
T	T	0.24
T	F	0.96
F	T	0.48
F	F	0.31

Exact inference is intractable for large, multiply connected networks so we may need to consider approximate inference methods based on **Monte Carlo** algorithms.

Monte Carlo algorithms are used to estimate quantities that are difficult to calculate exactly.

- generate many samples
- use statistics to estimate the quantity
- more samples = more accuracy



A **sample** corresponds to an instantiation of random variables.

Each sample should be generated from a known probability distribution (given by CPTs in the Bayesian network).

- nodes (variables) are taken in topological order
- the probability distribution is conditioned on the values already assigned to parents
- generate a sample value based on this distribution

function PRIOR-SAMPLE(bn) **returns** an event sampled from the prior specified by bn
  inputs: bn, a Bayesian network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$

  $\mathbf{x} \leftarrow$ an event with n elements
  foreach variable X_i **in** $X_1, \ldots, X_n$ **do**
    $\mathbf{x}[i] \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$
  return x

Let N be the number of samples and $N(x_1,\ldots,x_n)$ be the number of occurrences of event $x_1,\ldots,x_n$, then

$$P(x_1,\ldots,x_n) = \lim_{N\to\infty} (N(x_1,\ldots,x_n)/N)$$

However, we are looking for **P**(X | e)!

Rejection sampling

From all the generated samples, we will select only those consistent with the evidence **e** (other samples are rejected).

$$\mathbf{P}(X \mid e) \approx \mathbf{N}(X,e) / N(e)$$

Major weakness: **rejecting too many samples**

```
function REJECTION-SAMPLING(X, e, bn, N) returns an estimate of P(X|e)
  inputs: X, the query variable
          e, observed values for variables E
          bn, a Bayesian network
          N, the total number of samples to be generated
  local variables: N, a vector of counts for each value of X, initially zero

  for j = 1 to N do
     x ← PRIOR-SAMPLE(bn)
     if x is consistent with e then
        N[x] ← N[x]+1 where x is the value of X in x
  return NORMALIZE(N)
```

REJECTED

Likelihood weighting

Instead of rejecting inconsistent samples, it seems more efficient to generate only samples consistent with evidence **e**.

- Fix the values for the evidence variables **E** and sample only the non-evidence variables.
- The probability of obtaining a sample is $P(z,e) = \prod_i P(z_i \mid parents(z_i))$
- But this is not what we want! We miss $w(z,e) = \prod_j P(e_j \mid parents(e_j))$.
- Hence each sample is **weighted** as follows: $\mathbf{P}(X \mid e) \approx \alpha\, \mathbf{N}(X,e)\, w(X,e)$

Major weakness: **too small weights**

```
function LIKELIHOOD-WEIGHTING(X, e, bn, N) returns an estimate of P(X|e)
  inputs: X, the query variable
          e, observed values for variables E
          bn, a Bayesian network specifying joint distribution P(X1, ..., Xn)
          N, the total number of samples to be generated
  local variables: W, a vector of weighted counts for each value of X, initially zero

  for j = 1 to N do
     x, w ← WEIGHTED-SAMPLE(bn, e)
     W[x] ← W[x] + w where x is the value of X in x
  return NORMALIZE(W)
```

```
function WEIGHTED-SAMPLE(bn, e) returns an event and a weight

  w ← 1; x ← an event with n elements initialized from e
  foreach variable Xi in X1, ..., Xn do
     if Xi is an evidence variable with value xi in e
        then w ← w × P(Xi = xi | parents(Xi))
        else x[i] ← a random sample from P(Xi | parents(Xi))
  return x, w
```

Probability theory is a formal tool to handle **uncertainty**.

The **full joint probability distribution** specifies the probability of each complete assignment of values to random variables (possible worlds).

It is usually too large but **independence** relations between subsets of random variables allows us to factor it into smaller joint or conditional distributions.

Bayesian network is such a compact representation.

We can use it to answer **queries P(X|E)** about probability distributions of variables **X** under evidence **E**.

- exact methods (enumeration, variable elimination)
- approximate methods (rejection sampling, likelihood weighting)

Department of Theoretical Computer Science and Mathematical Logic
bartak@ktiml.mff.cuni.cz