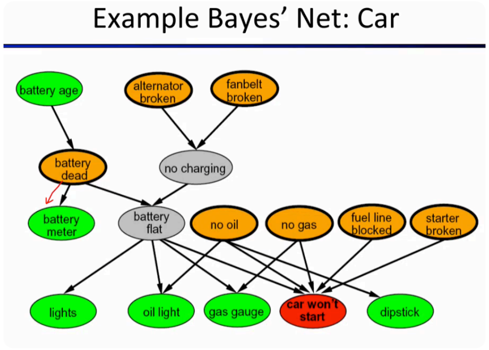
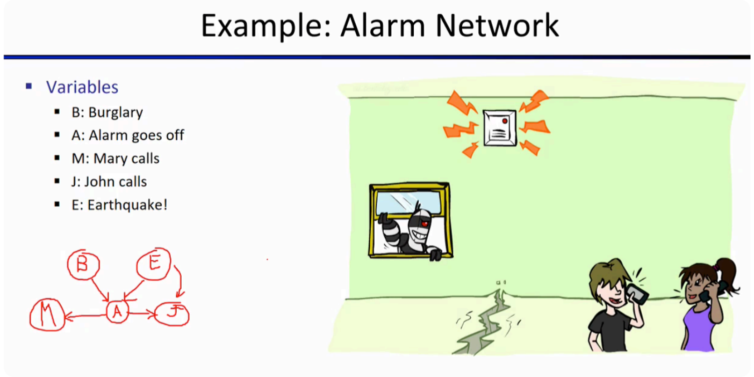
- Independence
  - 2 RV's indep. if their outcomes don't affect each other $X \perp\!\!\!\perp Y$: indep.
  - indep. iff $P(x, y) = P(x) P(y)$
    - for all possible values x, y
  - usually variables aren't indep., which allows us to do inference
    - can use independence as a modeling assumption (assume two variables are indep. = simplifies math)
    - **if variables are indep., you need FAR less parameters to specify their joint distribution**
      - $O(d * n)$ instead of $O(d^n)$ where d is the size of the domain of each of the n variables
  - conditional indep.
    - ex. P(catch, toothache, cavity) - probability of probe catching a cavity and the patient having a cavity and toothache
      - $P(catch \mid toothache, +cavity) = P(catch \mid +cavity)$ $X \perp\!\!\!\perp Y \mid Z$
      - $P(catch \mid toothache, -cavity) = P(catch \mid -cavity)$
        - probability of catching the cavity is indep. of the patient having a toothache GIVEN the presence or absence of a cavity
        - catching is conditionally indep. of patient feeling a toothache, given presence/absence cavity
          - catching is still related to presence/absence of toothache (if a patient has a toothache, the probability of catching a cavity would probably go up, but conditioned on the presence/absence of a cavity, it doesn't matter whether the patient feels a toothache or not, we catch the cavity with the same probability)
      - equivalent statements:
        - $P(toothache \mid catch, cavity) = P(toothache \mid cavity)$
        - $P(toothache, catch \mid cavity) = P(toothache \mid cavity) * P(catch \mid cavity)$
          - definition of indep., this time conditioned on cavity
    - we like conditional indep. because it comes up very frequently when modeling real world environ.
      - we make conditional indep. assumptions in our model
    - X is conditionally indep. of Y given Z iff:
      - $P(x, y \mid z) = P(x \mid z) P(y \mid z)$
        - this follows from the eq. below using chain rule
        - $P(x, y \mid z) = P(x \mid z) P(y \mid x, z) = P(x \mid z) * P(y \mid z)$
          - where the last equality follows from cond. indep. of X, Y
      - or $P(x \mid y, z) = P(x \mid z)$
      - "knowing Z, Y won't give us any additional information about X"
    - ex. fire, smoke, alarm
      - Smoke is conditionally indep. of alarm given there's a fire
- Conditional indep. and the chain rule
  - $P(X1, X2, ..., Xn) = P(X1) P(X2 \mid X1) -- P(Xn \mid X1, X2, ..., X_{n-1})$
    - representing the joint using conditionals is NOT more space efficient
    - the last conditional requires a table that is $O(d^n)$ and we also need all the other smaller tables too (so this is actually worse)
  - Conditional indep. let's us reduce these sizes if we are careful about our expansion
    - $P(fire, smoke, alarm) = P(fire) P(alarm \mid fire) P(smoke \mid alarm, fire)$
      - $= P(fire) P(alarm \mid fire) P(smoke \mid fire)$
      - we can remove conditional variables from the LAST term which has the largest table size, and this can drastically reduce the number of parameters needed if we can remove many conditional var.'s
    - $O(nd^k) < O(d^n)$ for $k < n$
- **Bayes' nets / graphical models** (represent them w/ graphs) - an efficient expression of a probabilistic model with all its conditional indep. assumptions
  - the advantage to this is that we can then express the full joint distribution of a model with far fewer parameters (a bunch of small tables rather than one gigantic exponentially-sized table)
  - problems with using full joint distributions in our model
    - size of jt distr. table is way too large for many variables
    - hard to estimate the jt distr. empirically for many variables (need exponential-sized data set, at least a couple of datapoints for each joint probability entry)
  - describe complex full joint distr. (models) with simple, local distr. (cond. probabilities)


Example Bayes' Net: Car

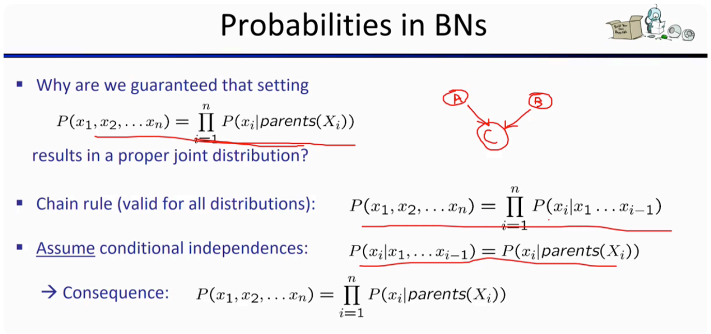
        - describe how variables interact locally
        - these local interactions can be chained together (chain rule) to give global, indirect interactions (full jt.)
- **Graphical model**:
    - modeling the relationships of variables (how they influence each other)
    - every variable is a node
        - can be assigned a value (observed variable, shaded) or unassigned (not-yet observed, not shaded)
    - edges represent interactions of variables (indicates "direct influence" b/t variables)
        - e.g. we draw an edge from a variable to another if we think there's a causal relationship
        - formally encodes conditional indep.
        - similar to CSP constraints
        - e.g. n indep. coin flips: n variables (one for each flip) with no edges between them (all indep.)
    - variables that don't share an edge are indep. in our model
        - won't be able to see how they affect each other because we assume they don't
    - for each variable, it's probability is conditioned on each of its parents (variables with edges to this one)
    - e.g. Alarm goes off => causes Mary and John to call
        - earthquake happens => John calls to check on you
        - depends on how you want to model the world



Example: Alarm Network

- Variables
    - B: Burglary
    - A: Alarm goes off
    - M: Mary calls
    - J: John calls
    - E: Earthquake!

- Bayes' nets semantics:
    - set of nodes, one per variable
    - is a DAG (directed, acyclic graph)
    - conditional distr. associated w/ each node X: P(X | x1, x2, ..., xn)
        - where x1, x2, ..., xn are X's parents
        - X is indep. of all other variables given its parents
        - this is called a CPT: conditional probability table
            - will describe a noisy "causal" process
            - this table does not sum to 1; sums to d^n where n is the number of parents



Probabilities in BNs

- Why are we guaranteed that setting

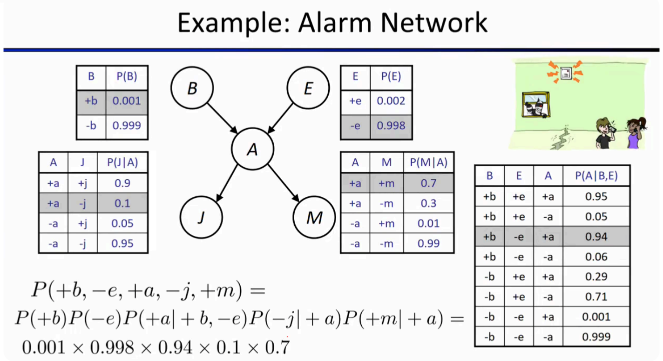
  $$P(x_1, x_2, \ldots x_n) = \prod_{i=1}^{n} P(x_i | parents(X_i))$$

  results in a proper joint distribution?

- Chain rule (valid for all distributions): $P(x_1, x_2, \ldots x_n) = \prod_{i=1}^{n} P(x_i | x_1 \ldots x_{i-1})$
- Assume conditional independences: $P(x_i | x_1, \ldots x_{i-1}) = P(x_i | parents(X_i))$

→ Consequence: $P(x_1, x_2, \ldots x_n) = \prod_{i=1}^{n} P(x_i | parents(X_i))$

    - essentially a Bayes' net is a topology (graph) + local conditional probabilities
    - implicitly encodes joint distr. in the small CPTs (chain rule, product of all the local CPTs)
        - if we assume conditional independences encoded by the graph, then the chain rule says we just need to multiply all the CPTs together (all the conditionally independent variables are by construction removed from our CPTs for us, i.e. all variables that aren't parents will not be included in the CPT)
        - choose a topological ordering of the DAG, and apply the chain rule



Example: Alarm Network

B	P(B)
+b	0.001
-b	0.999

E	P(E)
+e	0.002
-e	0.998

A	J	P(J|A)
+a	+j	0.9
+a	-j	0.1
-a	+j	0.05
-a	-j	0.95

A	M	P(M|A)
+a	+m	0.7
+a	-m	0.3
-a	+m	0.01
-a	-m	0.99

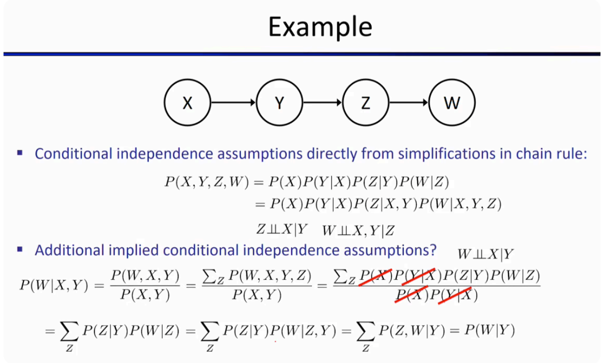
B	E	A	P(A|B,E)
+b	+e	+a	0.95
+b	+e	-a	0.05
+b	-e	+a	0.94
+b	-e	-a	0.06
-b	+e	+a	0.29
-b	+e	-a	0.71
-b	-e	+a	0.001
-b	-e	-a	0.999

$$P(+b, -e, +a, -j, +m) =$$
$$P(+b)P(-e)P(+a| + b, -e)P(-j| + a)P(+m| + a) =$$
$$0.001 \times 0.998 \times 0.94 \times 0.1 \times 0.7$$

    - we limit the possible joint distr.'s/real models we can represent, since our topology enforces a specific set of conditional probabilities
    - e.g.
        - Alarm has the biggest CPT because it has 2 parents (all variables have the same domain size = 2)

- **Causality**
    - having edges represent causality patterns is just a simplification (for construction, analyzing, etc.)
    - BNs don't have to be causal
        - if you reverse causal edges, the resulting joint distr. will still be the same!
        - you can draw edges showing correlated variables (no causal relationship, but intermediary variables not included in our BN)
            - e.g. rain -> drip, rain -> traffic, if we remove rain, we should draw an edge drip - traffic to show that they're correlated
    - topology may happened to encode causal structure but REALLY encodes conditional indep. assumptions
- Questions we can ask the BN:
    - inference: what is P(X | e)?
    - representation: what kind of distributions can the BN graph encode?
        - some information must be lost in the BN since it "compresses" the joint distribution
    - modeling: what BN is most appropriate for a given domain

- **Size of a Bayes' net**
    - Assume boolean variables, distribution represented as a table
        - a joint distribution over N variables has size 2^n
        - an N-node net, each node has k parents => size O(N * 2^(k + 1))
            - k+1 because your CPTs have to account for all of the parents' and child's values
            - N because we have N variables
            - if we limit the number of parents, our BN can give huge space savings
    - easier to elicit local CPTs (from experts) when constructing model than trying to figure out one gigantic joint
    - also faster to answer queries using BN
- **Independence in Bayes' Nets**
    - BN is able to define a distribution (joint) compactly because it makes conditional indep. assumptions
        - each node is conditionally indep. of all other nodes GIVEN its parents
    - Beyond the obvious conditional indep. encoded in parent-child edges, we have implicit cond. indep. assumptions:
        - Y gives us all the info we need to get the probability of Z which in turn gives us the probability of W indep. of what info X provides
            - this occurs even though Y does not have a direct arrow to W, though W is "downstream of Y and X"
    - Are 2 nodes independent given some other node/ evidence? if yes, prove with algebra (tedious); if no, prove by a counterexample (just link X = Y to show that X and Y are not conditionally indep.)



Example

- Conditional independence assumptions directly from simplifications in chain rule:

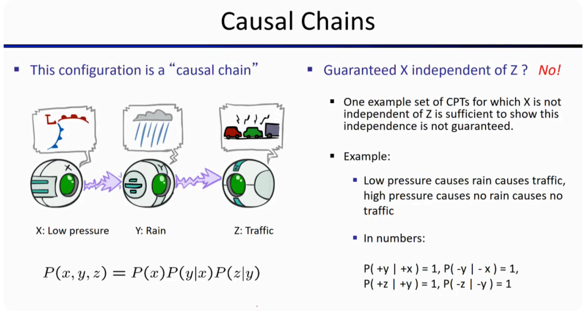
$$P(X,Y,Z,W) = P(X)P(Y|X)P(Z|Y)P(W|Z)$$
$$= P(X)P(Y|X)P(Z|X,Y)P(W|X,Y,Z)$$
$$Z \perp\!\!\!\perp X|Y \quad W \perp\!\!\!\perp X,Y|Z$$

- Additional implied conditional independence assumptions? $W \perp\!\!\!\perp X|Y$

$$P(W|X,Y) = \frac{P(W,X,Y)}{P(X,Y)} = \frac{\sum_Z P(W,X,Y,Z)}{P(X,Y)} = \frac{\sum_Z \cancel{P(X)}\cancel{P(Y|X)}P(Z|Y)P(W|Z)}{\cancel{P(X)}\cancel{P(Y|X)}}$$

$$= \sum_Z P(Z|Y)P(W|Z) = \sum_Z P(Z|Y)P(W|Z,Y) = \sum_Z P(Z,W|Y) = P(W|Y)$$

- **D-separation** provides us with a method to determine whether 2 variables are *guaranteed* conditionally indep. by graphical analysis (instead of formal algebra or finding a counterexample, which can be tedious)
    - a condition / alg. for answering whether 2 variables are conditional indep. given another
    - first we'll study indep. properties for triples, then we'll generalize to more complex cases in terms of constituent triples
    - There are only three configurations of triples:
        - **Causal chain**
            - Z indep. of X given Y is always true
                - P(z | x, y) = P(x, y, z) / P(x, y)
                    - = P(x) P(y | x) P(z | y) / P(x) P(y | x) = P(z | y)
                - Once we get to Y, we can determine anything downstream without needing anything from upstream
                - observed Y blocks influence of anything upstream (assertive boss = no need to listen to anybody higher)
            - Z not indep. of X without condition



Causal Chains

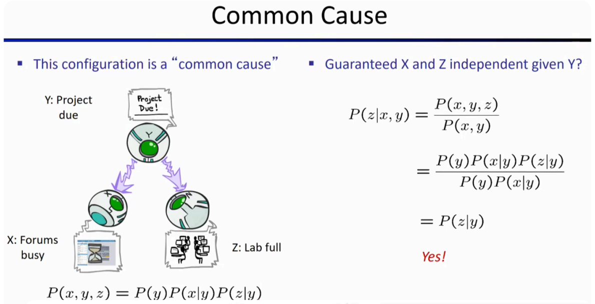
- This configuration is a "causal chain"

$$P(x,y,z) = P(x)P(y|x)P(z|y)$$

- Guaranteed X independent of Z ? *No!*
    - One example set of CPTs for which X is not independent of Z is sufficient to show this independence is not guaranteed.
    - Example:
        - Low pressure causes rain causes traffic, high pressure causes no rain causes no traffic
        - In numbers:

P(+y | +x) = 1, P(-y | - x) = 1,
P(+z | +y) = 1, P(-z | -y) = 1

      - make Z = Y = X
  - **Common cause**
    - Z indep. of X given Y is always true
      - same proof as above
      - just need to look upstream, sibling tributaries don't affect this one
      - active boss means suborb. don't have any ability to influence each other
    - Z not indep. of X without condition
      - can use same example as causal chains
      - basically if forums busy = more likely there's a project due = more likely labs are also full
      - siblings influence each other thru passive parent/boss
  - **Common effect**
    - X and Y are indep. because they don't affect each other
      - unobserved Z blocks their influence
      - inactive child = can't influence each other
    - X and Y are NOT indep. GIVEN Z
      - if there's traffic, and it's not raining, then there's probably a ball game (something has to explain the traffic)
      - the X, Y do influence each THRU an observed Z (they're connected thru Z)
      - influence thru manifested/hyperactive child/subordinate



Common Cause

- This configuration is a “common cause”
- Guaranteed X and Z independent given Y?

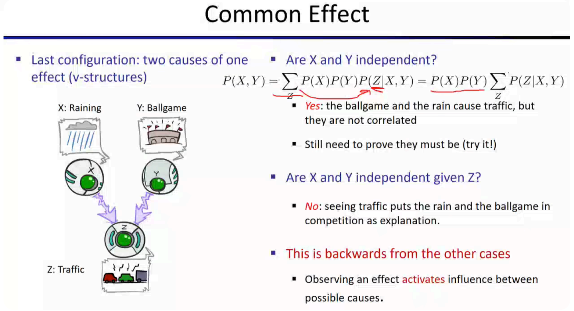
$$P(x,y,z) = P(y)P(x|y)P(z|y)$$

$$P(z|x,y) = \frac{P(x,y,z)}{P(x,y)}$$

$$= \frac{P(y)P(x|y)P(z|y)}{P(y)P(x|y)}$$

$$= P(z|y)$$

Yes!



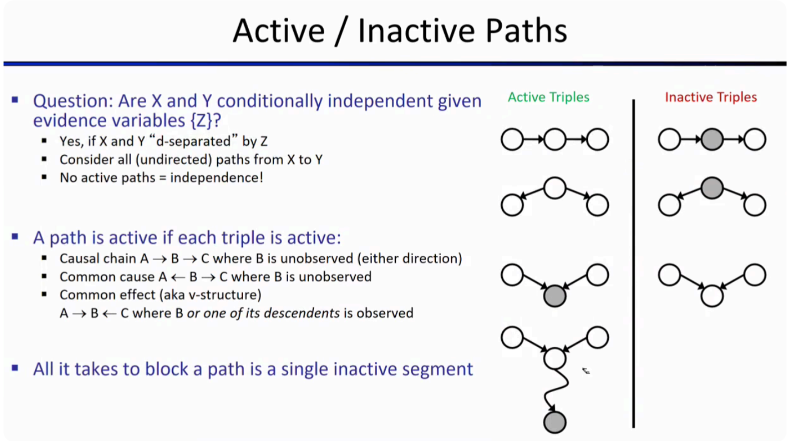
Common Effect

- Last configuration: two causes of one effect (v-structures)
- Are X and Y independent?

$$P(X,Y) = \sum_Z P(X)P(Y)P(Z|X,Y) = P(X)P(Y)\sum_Z P(Z|X,Y)$$

  - *Yes*: the ballgame and the rain cause traffic, but they are not correlated
  - Still need to prove they must be (try it!)
- Are X and Y independent given Z?
  - *No*: seeing traffic puts the rain and the ballgame in competition as explanation.
- This is backwards from the other cases
  - Observing an effect activates influence between possible causes.

- Just by applying these 3 templates, we can analyze the guaranteed conditional indep. relationships of any 2 variables in the BN graph (whether 2 variables are *guaranteed* to be cond. indep.)
  - any descendant gives you information about the ancestor (e.g. last active triple below)

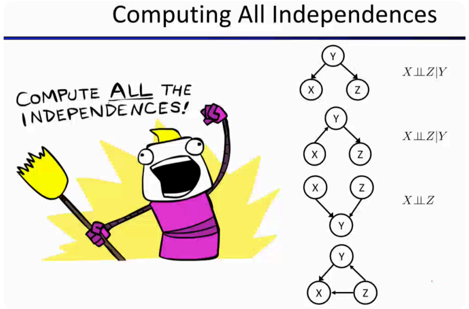


Active / Inactive Paths

- Question: Are X and Y conditionally independent given evidence variables {Z}?
  - Yes, if X and Y “d-separated” by Z
  - Consider all (undirected) paths from X to Y
  - No active paths = independence!
- A path is active if each triple is active:
  - Causal chain A → B → C where B is unobserved (either direction)
  - Common cause A ← B → C where B is unobserved
  - Common effect (aka v-structure) A → B ← C where B *or one of its descendents* is observed
- All it takes to block a path is a single inactive segment

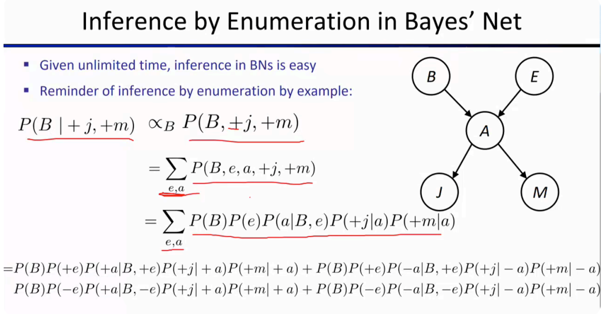
  - X and Y are "d-separated" if all paths between them are inactive (no flow of influence = they don't affect each other and are indep.)
    - if any path between them is active, cond. indep. is NOT guaranteed
      - if influence can flow on any path
    - if ALL paths between them are inactive, cond. indep. IS guaranteed
      - look for blockages of influence
    - paths are going to be made of constituent triples (any inactive triple makes the path inactive)
  - shaded = observed variables, unshaded = unobserved variables
- Given a BN graph, we can run d-separation alg. to build complete list of conditional indep.'s
  - this list then tells us the set of probability distr. that this BN can represent
  - e.g. computing all independences
    - the 1st and 2nd BN actually encode the same set distributions (i.e. those where X _||_ Z | Y)

                - flipping the arrows doesn't NECESSARILY change the distributions the BN can represent
            - the 3rd BN encodes a different set of possible distributions
                - flipping the arrows CAN change the distributions the BN can represent (flipped Y-Z edge from 2nd BN)
            - the 4th BN has no conditional indep., no info. is "compressed" => it can represent all possible distributions of 3 variables


Computing All Independences

COMPUTE ALL THE INDEPENDENCES!
Y
X
Z
$X \perp\!\!\!\perp Z | Y$
Y
X
Z
$X \perp\!\!\!\perp Z | Y$
X
Z
Y
$X \perp\!\!\!\perp Z$
Y
X
Z

- **Inference** - how we answer questions about query variables given evidence
    - inference by enumeration (see last note): get P(query | evidence), marginalizing out hidden variables
        - select entries consistent with evidence in big jt table
        - marginalize out H vars
        - normalize to get a valid conditional distribution for query | evidence
        - IBE gets hairy and slow for large BN (requires us to build full joint); instead we should eliminate hidden variables before we build full joint to reduce size of the table before we compute
            - **variable elimination:** interleave joining and marginalizing (still NP hard, but usually faster than IBE)


Inference by Enumeration in Bayes' Net

- Given unlimited time, inference in BNs is easy
- Reminder of inference by enumeration by example:

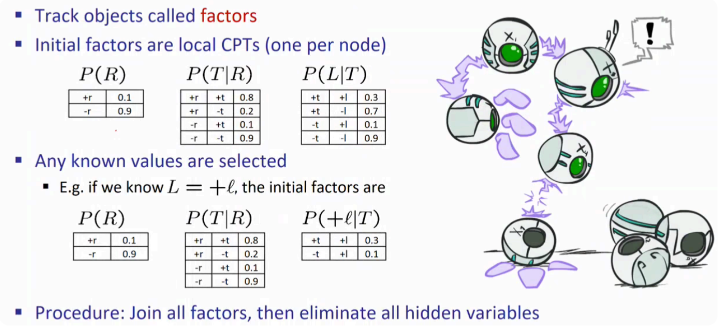
$$P(B \mid +j, +m) \;\propto_B\; P(B, +j, +m)$$

$$= \sum_{e,a} P(B, e, a, +j, +m)$$

$$= \sum_{e,a} P(B)P(e)P(a|B,e)P(+j|a)P(+m|a)$$

$$=P(B)P(+e)P(+a|B,+e)P(+j|+a)P(+m|+a) + P(B)P(+e)P(-a|B,+e)P(+j|-a)P(+m|-a)$$
$$P(B)P(-e)P(+a|B,-e)P(+j|+a)P(+m|+a) + P(B)P(-e)P(-a|B,-e)P(+j|-a)P(+m|-a)$$

    - most likely explanation: argmax_q P(Q = q | evidence)
    - **Factors:** any probability table
        - joint distribution - P(X, Y)
            - sums to 1, over all possible values x, y
        - selected joint - a slice of the joint distribution
            - e.g. P(x, Y): fixed x, over all possible values of y
            - sums to P(x)
        - single conditional - P(Y | x)
            - fixed x, over all y
            - sums to 1
        - family of conditionals - P(X | Y)
            - multiple conditionals over all x for a given y (over all y)
            - sums to |Y|
        - specified family - P(y | X)
            - P(y | x) for a fixed y, over all x
            - could sum to anything
    - number of "capital letters" = dimensionality of factor table (|P(x, Y)| = domain size |Y|)
        - multiply the cardinality of the capital variables to get the dimensionality of the table


- Track objects called factors
- Initial factors are local CPTs (one per node)

$P(R)$

+r	0.1
-r	0.9

$P(T|R)$

+r	+t	0.8
+r	-t	0.2
-r	+t	0.1
-r	-t	0.9

$P(L|T)$

+t	+l	0.3
+t	-l	0.7
-t	+l	0.1
-t	-l	0.9

- Any known values are selected
    - E.g. if we know $L = +\ell$, the initial factors are

$P(R)$

+r	0.1
-r	0.9

$P(T|R)$

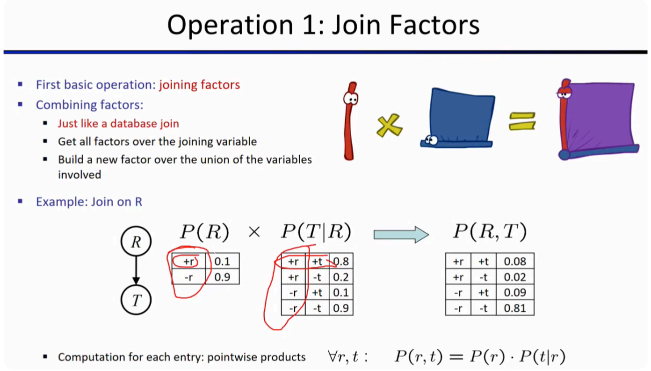
+r	+t	0.8
+r	-t	0.2
-r	+t	0.1
-r	-t	0.9

$P(+\ell|T)$

+t	+l	0.3
-t	+l	0.1

- Procedure: Join all factors, then eliminate all hidden variables

    - **Procedure for IBE using factors**
        - want to keep track our factor objects
        - initial factors are local CPTs (one per node)
        - select entries in factors consistent with evidence
        - **join all factors and eliminate hidden vars**
            - join on a specific variable
            - build a new factor by joining (database-wise) all tables involving the joining variable
            - point wise products of entries that share the same value for a particular


Operation 1: Join Factors

- First basic operation: joining factors
- Combining factors:
    - Just like a database join
    - Get all factors over the joining variable
    - Build a new factor over the union of the variables involved
- Example: Join on R

$P(R) \times P(T|R)$ → $P(R,T)$

$P(R)$

+r	0.1
-r	0.9

$P(T|R)$

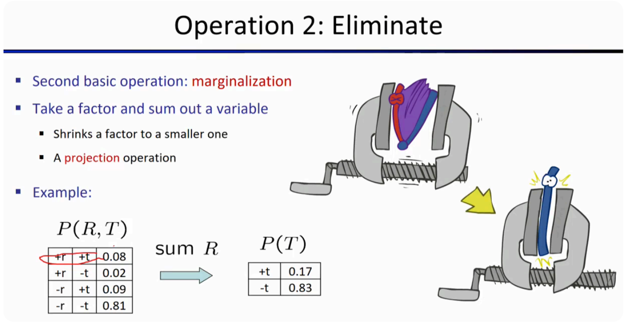
+r	+t	0.8
+r	-t	0.2
-r	+t	0.1
-r	-t	0.9

$P(R,T)$

+r	+t	0.08
+r	-t	0.02
-r	+t	0.09
-r	-t	0.81

- Computation for each entry: pointwise products $\forall r, t: \quad P(r,t) = P(r) \cdot P(t|r)$

value of the joining variable (e.g. R in the ex. in the figure)
    - won't always result in a joint distribution factor
- Eliminating hidden vars means marginalizing them out: the sum of the table remains unchanged



Operation 2: Eliminate

- Second basic operation: marginalization
- Take a factor and sum out a variable
    - Shrinks a factor to a smaller one
    - A projection operation
- Example:

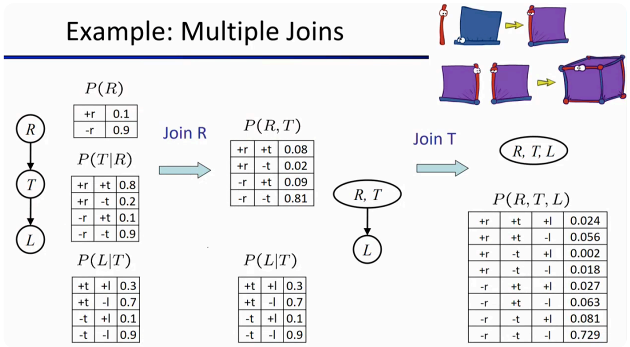
$P(R,T)$

+r	+t	0.08
+r	-t	0.02
-r	+t	0.09
-r	-t	0.81

sum R

$P(T)$

+t	0.17
-t	0.83



Example: Multiple Joins

$P(R)$

+r	0.1
-r	0.9

$P(T|R)$

+r	+t	0.8
+r	-t	0.2
-r	+t	0.1
-r	-t	0.9

$P(L|T)$

+t	+l	0.3
+t	-l	0.7
-t	+l	0.1
-t	-l	0.9

$P(R,T)$

+r	+t	0.08
+r	-t	0.02
-r	+t	0.09
-r	-t	0.81

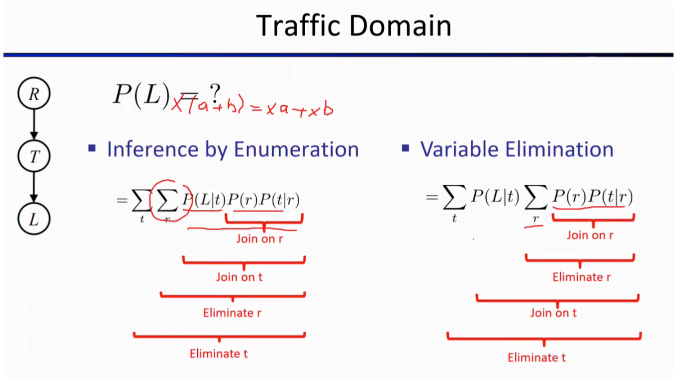
$P(L|T)$

+t	+l	0.3
+t	-l	0.7
-t	+l	0.1
-t	-l	0.9

$P(R,T,L)$

+r	+t	+l	0.024
+r	+t	-l	0.056
+r	-t	+l	0.002
+r	-t	-l	0.018
-r	+t	+l	0.027
-r	+t	-l	0.063
-r	-t	+l	0.081
-r	-t	-l	0.729

- IBE is we *first* join all our CPT factors and *then* eliminate all hidden vars by marginalizing
    - works, but inefficient b/c joining everything gives a giant, potentially exponentially sized table
    - variable elimination will let us pare down our factors along the way so we have less entries to keep track of at each step (minimum entries kept track of at each time)
    - In the example of traffic domain in figure right, we recognize that P(L | t) can be pulled out of the sum over R, since it doesn't depend on R (algebraically same)
        - this means at any point (assuming all variables boolean), the largest factor we need to keep track of in variable elimination is 2^2 = 4 instead of 2^3 = 8 (as with IBE)



Traffic Domain

$P(L) = ?$ x(a+b) = xa+xb

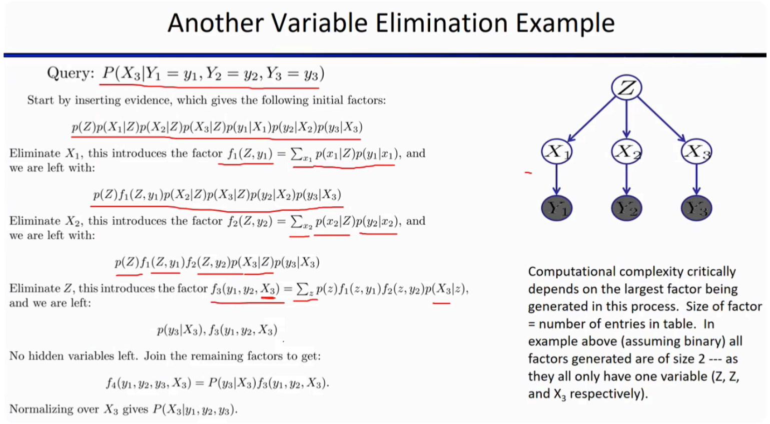
- Inference by Enumeration

$$= \sum_t \sum_r \underbrace{\underbrace{\underbrace{\underbrace{P(L|t)\underbrace{P(r)P(t|r)}_{\text{Join on r}}}_{\text{Join on t}}}_{\text{Eliminate r}}}_{\text{Eliminate t}}}$$

- Variable Elimination

$$= \sum_t P(L|t) \sum_r \underbrace{\underbrace{\underbrace{\underbrace{P(r)P(t|r)}_{\text{Join on r}}}_{\text{Eliminate r}}}_{\text{Join on t}}}_{\text{Eliminate t}}$$

- **Variable elimination (VE)** - *marginalizing early*
    - join on a variable R (=> now we have exactly 1 factor that contains R), and then sum R out immediately (eliminate it from any factor you have)
    - if we have evidence to begin with, we can just throw out entries in our initial factors (local CPTs) that don't agree with our evidence
        - else if we have no evidence, just keep all initial CPT factors unchanged
    - <u>Formal Procedure:</u>
        - want P(Q | E)
        - Start with initial factors (local CPTs, pared down by given evidence - select only entries consistent w/ evidence)
        - While still hidden variables:
            - pick a hidden var H
            - join all factors on H (all factors mentioning H)
            - eliminate (marginalize/ sum out) H
        - join all remaining factors and normalize
    - <u>How you pick the order of joining/marginalizing out hidden var's matters for efficiency</u>
        - *complexity* = ***largest factor*** *formed in the procedure*
        - it turns out we want to eliminate X_1, ..., X_{n-1} (hidden variables that we don't care about) before Z, because we keep a bunch of tiny tables at each step
            - if we eliminate Z first, we need to join n+1 tables (n



Another Variable Elimination Example

Query: $P(X_3|Y_1 = y_1, Y_2 = y_2, Y_3 = y_3)$

Start by inserting evidence, which gives the following initial factors:

$$p(Z)p(X_1|Z)p(X_2|Z)p(X_3|Z)p(y_1|X_1)p(y_2|X_2)p(y_3|X_3)$$

Eliminate X_1, this introduces the factor $f_1(Z, y_1) = \sum_{x_1} p(x_1|Z)p(y_1|x_1)$, and we are left with:

$$p(Z)f_1(Z, y_1)p(X_2|Z)p(X_3|Z)p(y_2|X_2)p(y_3|X_3)$$

Eliminate X_2, this introduces the factor $f_2(Z, y_2) = \sum_{x_2} p(x_2|Z)p(y_2|x_2)$, and we are left with:

$$p(Z)f_1(Z, y_1)f_2(Z, y_2)p(X_3|Z)p(y_3|X_3)$$

Eliminate Z, this introduces the factor $f_3(y_1, y_2, X_3) = \sum_z p(z)f_1(z, y_1)f_2(z, y_2)p(X_3|z)$, and we are left:

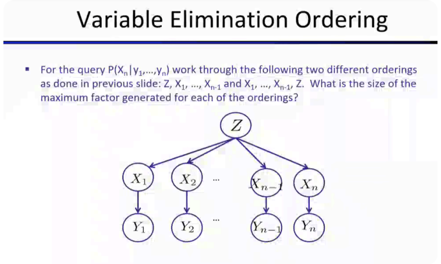
$$p(y_3|X_3), f_3(y_1, y_2, X_3)$$

No hidden variables left. Join the remaining factors to get:

$$f_4(y_1, y_2, y_3, X_3) = P(y_3|X_3)f_3(y_1, y_2, X_3).$$

Normalizing over X_3 gives $P(X_3|y_1, y_2, y_3)$.

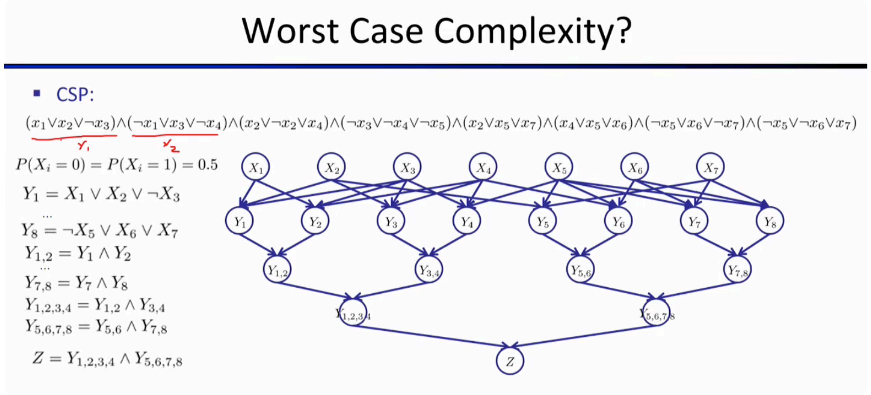
Computational complexity critically depends on the largest factor being generated in this process. Size of factor = number of entries in table. In example above (assuming binary) all factors generated are of size 2 --- as they all only have one variable (Z, Z, and X_3 respectively).



Variable Elimination Ordering

- For the query $P(X_n|y_1,\dots,y_n)$ work through the following two different orderings as done in previous slide: Z, X_1, …, X_{n-1} and X_1, …, X_{n-1}, Z. What is the size of the maximum factor generated for each of the orderings?

      conditional CPTs P(X_i | Z) and then P(Z)), each with 2 entries = a factor of size 2^(n+1) entries, we eliminate Z and get a table/factor of size 2^n
        - exponential = bad complexity
      - if we eliminate each of the X_i first (except X_n, our query), we join 2 tables each time: P(X_i | Z) P(y_i | X_i); since y_i fixed, the result of these joins is a 2^2 factor; we eliminate X_i to get a factor f(y_i, Z) of size 2; at the very end we join all the factors f(y_i, Z), P(Z), P(X_n | Z), P(y_n | X_n) on Z, and eliminate Z, then join on X_n (y_i fixed so this is last join is |Z| * |X_n| = 2^2)
        - overall we needed to keep track of a max of (n * 2^2) entries total at any step (max is initial local CPTs)
        - largest factor formed = 2^2
    - there is not guaranteed an ordering that only results in small factors (can be as bad as constructing full joint distribution)
- **Inference using BN is NP hard**
  - 3 SAT reduces to inference, meaning inference can be used to solve 3 SAT and is at least as hard as 3SAT (and we know 3 SAT is NP hard)
  - if we can do inference on Z, i.e. find P(Z | X_1, ..., X_n), we can find an assignment for the X_i's that satisfies all the triplets in polynomial time



Worst Case Complexity?

- CSP:

$$\underbrace{(x_1 \vee x_2 \vee \neg x_3)}_{Y_1} \wedge \underbrace{(\neg x_1 \vee x_3 \vee \neg x_4)}_{Y_2} \wedge (x_2 \vee \neg x_2 \vee x_4) \wedge (\neg x_3 \vee \neg x_4 \vee \neg x_5) \wedge (x_2 \vee x_5 \vee x_7) \wedge (x_4 \vee x_5 \vee x_6) \wedge (\neg x_5 \vee x_6 \vee \neg x_7) \wedge (\neg x_5 \vee \neg x_6 \vee x_7)$$

$P(X_i = 0) = P(X_i = 1) = 0.5$

$Y_1 = X_1 \vee X_2 \vee \neg X_3$

...

$Y_8 = \neg X_5 \vee X_6 \vee X_7$

$Y_{1,2} = Y_1 \wedge Y_2$

...

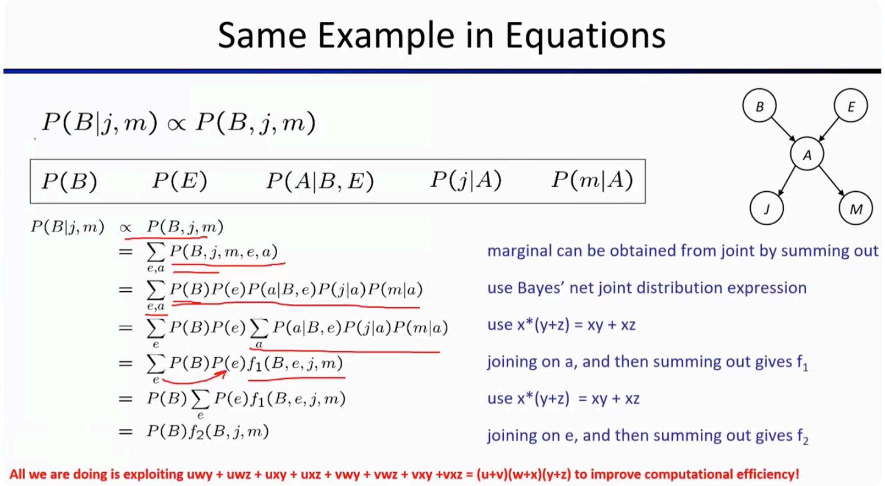
$Y_{7,8} = Y_7 \wedge Y_8$

$Y_{1,2,3,4} = Y_{1,2} \wedge Y_{3,4}$

$Y_{5,6,7,8} = Y_{5,6} \wedge Y_{7,8}$

$Z = Y_{1,2,3,4} \wedge Y_{5,6,7,8}$

  - Set up Bayes Net such that Z = true iff the 3SAT problem can be satisfied
    - Query is P(Z | X_i's), want to find if there is some combination of X_i's such that Z = true
      - i.e. is P(Z = T | X_i's) > 0?
    - Y_i's are the individual OR clauses of the X_i's (Y_i depends on the clause being satisfied), Y_(i, j,...) are the AND's of the clauses
    - Therefore if we can do inference on this Bayes' net in < O(exp.) we can solve 3SAT in < O(exp.) [not possible as far as we know]
  - Shows that inference via variable elimination is still bounded above by exponential time, but in many cases it can do much better
- Alarm network example:



Same Example in Equations

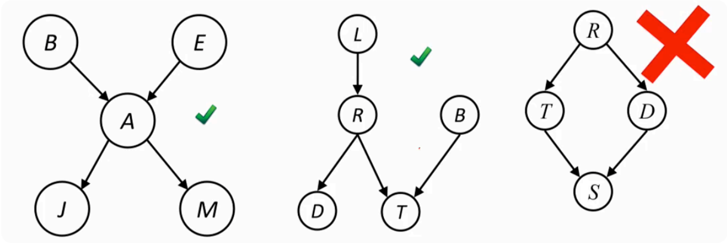
$$P(B|j,m) \propto P(B,j,m)$$

$\boxed{P(B) \quad P(E) \quad P(A|B,E) \quad P(j|A) \quad P(m|A)}$

$$\begin{aligned}
P(B|j,m) &\propto \underline{P(B,j,m)} \\
&= \sum_{e,a} \underline{P(B,j,m,e,a)} && \text{marginal can be obtained from joint by summing out} \\
&= \sum_{e,a} \underline{P(B)P(e)P(a|B,e)P(j|a)P(m|a)} && \text{use Bayes' net joint distribution expression} \\
&= \sum_{e} P(B)P(e) \underline{\sum_{a} P(a|B,e)P(j|a)P(m|a)} && \text{use x*(y+z) = xy + xz} \\
&= \sum_{e} P(B)P(e)\underline{f_1(B,e,j,m)} && \text{joining on a, and then summing out gives } f_1 \\
&= P(B)\sum_{e} P(e)f_1(B,e,j,m) && \text{use x*(y+z) = xy + xz} \\
&= P(B)f_2(B,j,m) && \text{joining on e, and then summing out gives } f_2
\end{aligned}$$

All we are doing is exploiting uwy + uwz + uxy + uxz + vwy + vwz + vxy +vxz = (u+v)(w+x)(y+z) to improve computational efficiency!

- **Polytrees** - type of Bayes' net for which inference is always easy/efficient
  - a polytree is a directed graph with no UNDIRECTED cycles (BN has no directed cycles since its a DAG)
    - can always find an efficient ordering for VE



- How do you order your variables?
  - search problem - search over all possible orderings and minimize size of largest factor
    - in general this turns out to be NP hard to
  - some good heuristics exist for finding a good ordering
    - smallest factor: pick the variable that will result in the *smallest factor* (not guaranteed to work well b/c myopic)
      - works on the Z - X_i's, Y_i's e
    - min-neighbors: VE on variables that appear in fewest factors (least number of domains to multiply, but domains can still be large)
    - min-weight: remove variables that cause fewest number of other variables to appear in multiple factors (joining on this variable will create a new CPT with other variables)