CS 295A/395D: Artificial Intelligence

Queries & Partial Observability

Prof. Emma Tosch

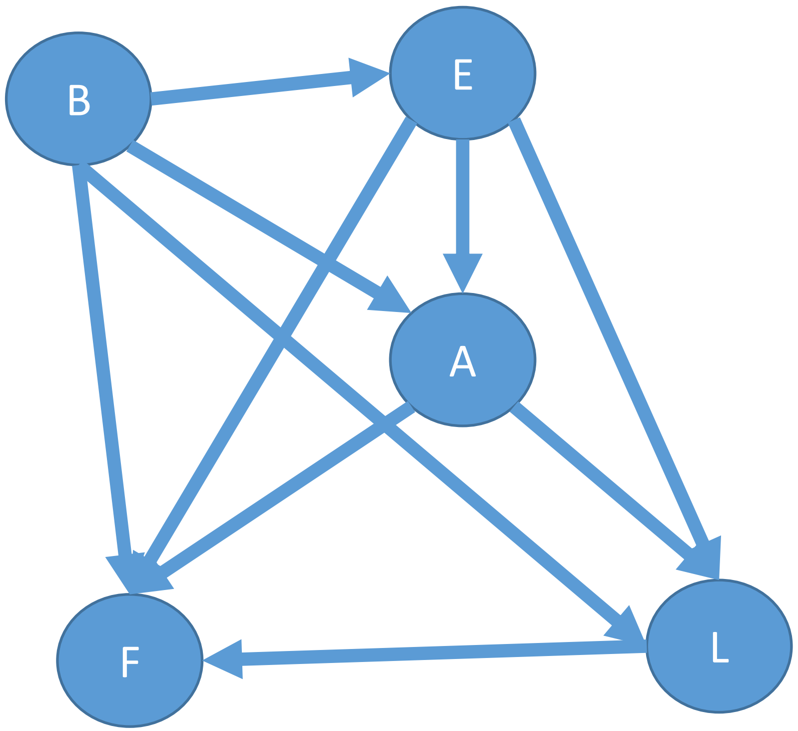
18 March 2022

The University of Vermont

Agenda

- New student hours: daily 10:30-noon, starting next week
  - Added class to my Teams calendar
  - May sometimes need to cut short
- Starting blogging again
- *Recap*: Bayes Nets for representing uncertain events
- *New*: **Using** Bayes Nets via querying
- *New*: Partial observability
- *New*: **Learning** Bayes Nets from data

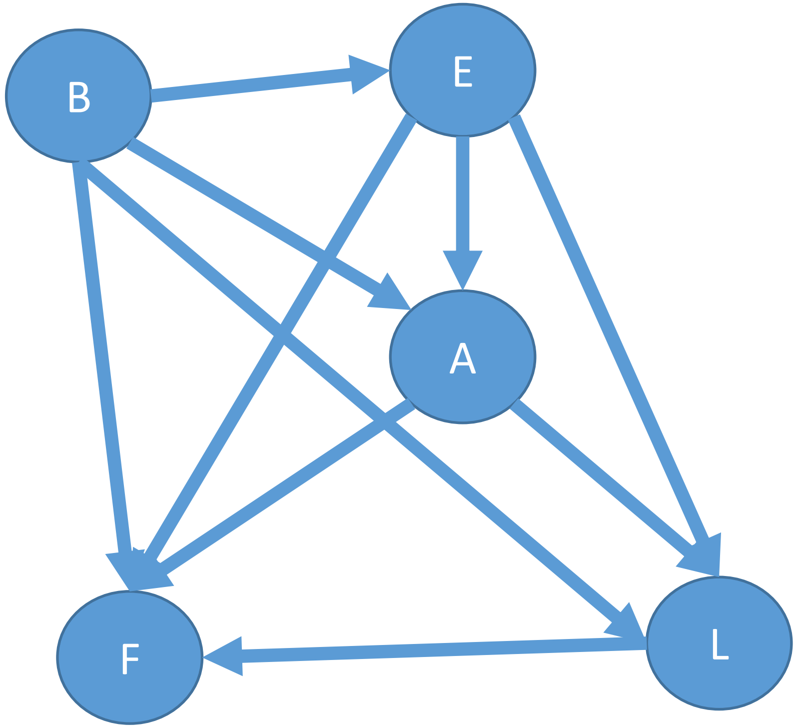
Recall: Factorization



Any joint probability distribution can be factorized using basic rules of probability:

$$P(B, E, A, F, L) = P(F \mid L, A, E, B) \times P(L \mid A, E, B) \times P(A \mid E, B) \times P(E \mid B) \times P(B)$$

Recall: Factorization



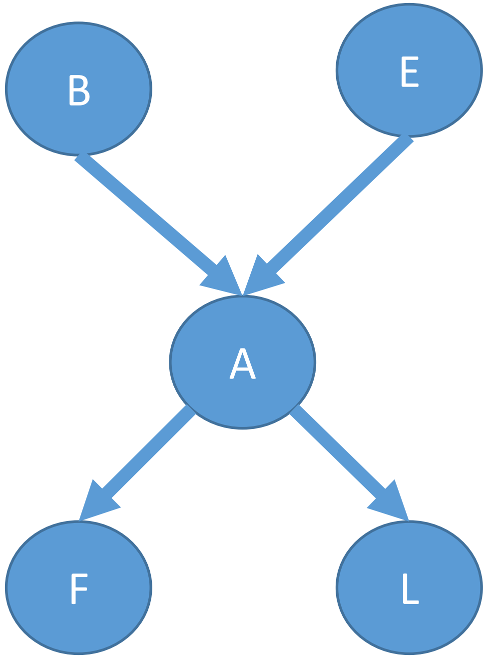
We can *draw* any factorization as a DAG where each node corresponds to a conditional probability distribution.

For discrete variables, this is a table:

Not strictly needed

L	B	E	A
l_1, p_1	b_1	e_1	a_1
l_2, p2	b_1	e_1	a_1
…	…	…	…
l_n, $1 – (p_1, … p_{n-1})$	b_1	e_1	a_1
l_1, p_1	b_1	e_1	a_2
…	…	…	..

Recap: Bayes Nets as factorization



1. Reverse topologically order the nodes, e.g.
   1. F, L, A, B, E **or**
   2. L, F, A, B, E, etc.

2. Factorize joint distribution using graph semantics of $\mathcal{G} =< \mathcal{V}, \mathcal{E} >, \quad \mathcal{V} = \{V_1, \ldots Vn\}$:

$$P(V_1, \ldots Vn) = \prod P(V_i | Parents(V_i))$$

here, P(B, E, A, F, L) = P(F | A)P(L | A)P(A | B, E)P(B)P(E)

Quasi-Recap: Learning Bayes Nets (In practice)

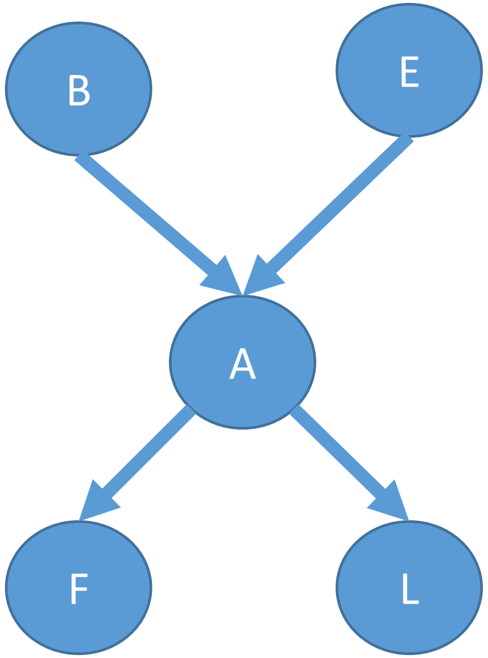
- Naïve approach: sample over a representative period, compute empirical probabilities of each possible event
  - **Problem**: some joint events are very low probability
- More practical approach: encode background knowledge in a DAG
  - Use existing estimates for model parameters (here, probabilities of discrete events)
  - Targeted sampling, other population distributions as beliefs
  - Directly encode probabilities as beliefs
    - May want to encode multiple hypotheses for parameter values (more an ML topic)

Without Bayes Nets: arbitrary queries are inefficient

Example arbitrary queries on P(X, Y, Z, W). How to compute:

- P(W) ?
- P(X, Y, Z | W)?
- P(X, Y | Z, W)?

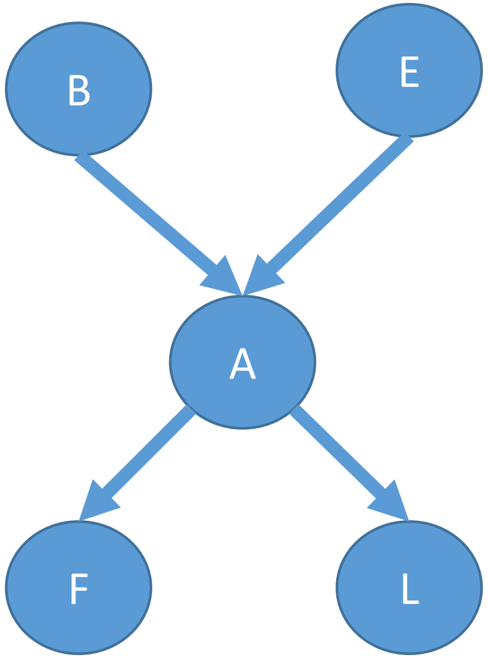
Recap: Bayes Nets are an *efficient* encoding



Example queries on P(B, E, A, F, L):

1. Example 1: P(B | E)
2. Example 2: P(F | L)
3. Example 3: P(F | L, A)

Recap: Bayes Nets encode *independences*



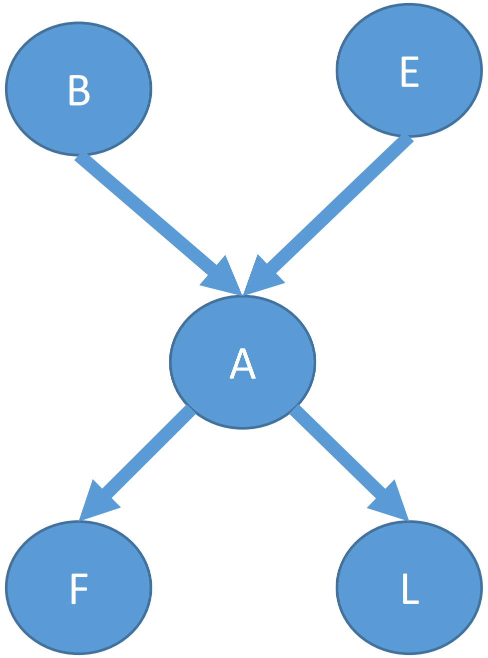
1. Edges denote *possible dependences*
2. Lack of *path* denotes *definite* independence

For independent events,

do not need to marginalize

if we have the graph!

d-separation



First…terminology:

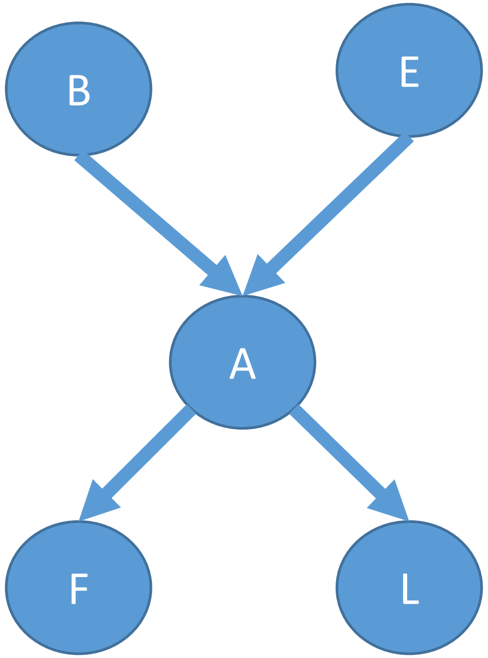
D-separation is defined in terms of:

1. Undirected paths through G
2. Substructures of G (directed)

Substructures:

- Chain (B->A->L)
- Fork (F <- A -> L)
- Collider (B -> A <- E)

d-separation



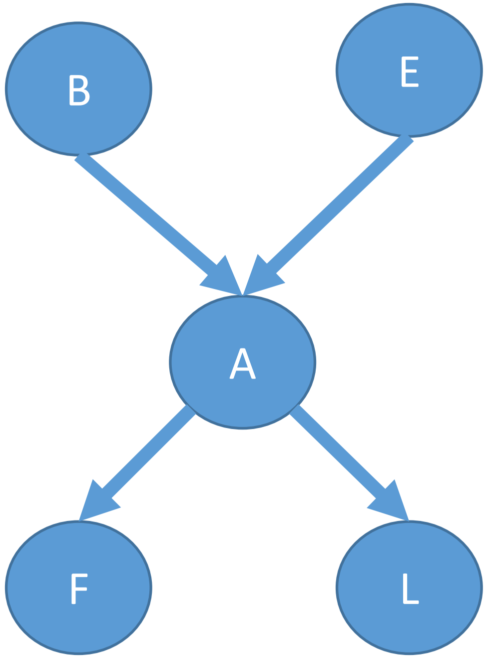
Classical definition (Pearl):

A set Z is said to d-separate X from Y iff Z blocks every path from a node in X to a node in Y.

A path p is blocked by Z iff:

1. p contains a chain i -> m -> j or a fork I <- m -> j such that m is in Z, or
2. p contains a collider i -> m <- j such that m is NOT in Z and *no descendant of m* is in Z.

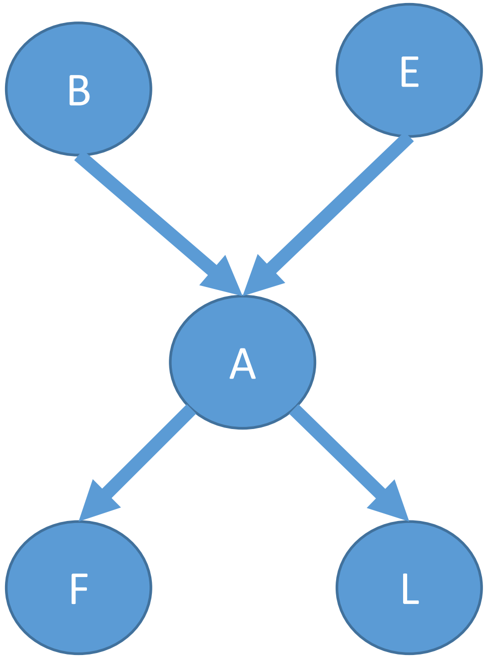
d-separation



Is B independent of E?

1. Z is empty
2. Find all paths from B to E.
3. For each path:
    1. Chain? If yes, are all the intermediate nodes in Z?
    2. Fork? If yes, are all the intermediate nodes in Z?
    3. Collider? If yes, are all the intermediate nodes AND their descendents NOT in Z?

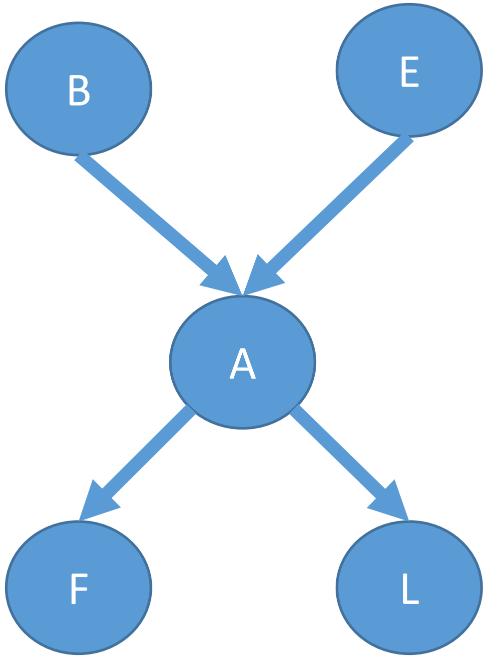
d-separation



Is F independent of L?

1. Z is empty
2. Find all paths from F to L.
3. For each path:
    1. Chain? If yes, are all the intermediate nodes in Z?
    2. Fork? If yes, are all the intermediate nodes in Z?
    3. Collider? If yes, are all the intermediate nodes AND their descendents NOT in Z?

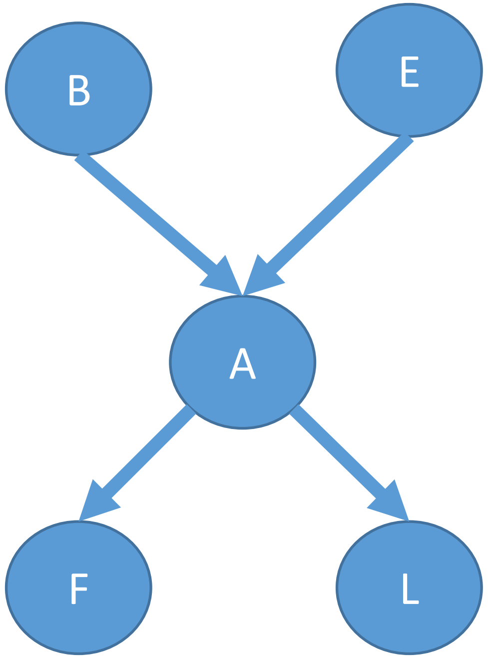
d-separation



Is F independent of L given A?

1. Z = {A}
2. Find all paths from F to L.
3. For each path:
    1. Chain? If yes, are all the intermediate nodes in Z?
    2. Fork? If yes, are all the intermediate nodes in Z?
    3. Collider? If yes, are all the intermediate nodes AND their descendants NOT in Z?

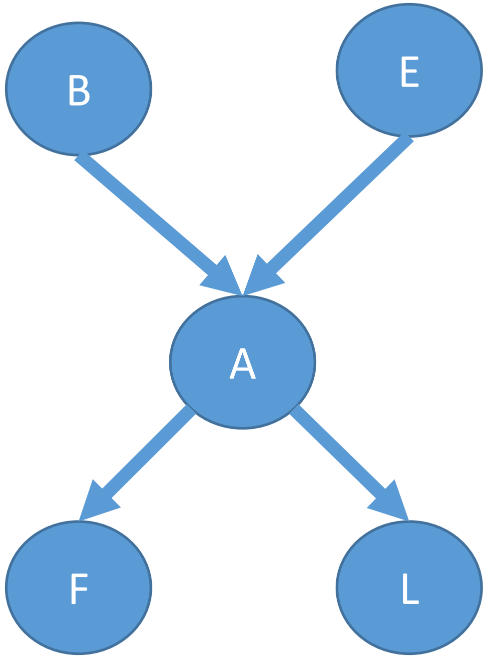
d-separation



Is B independent of E given L?

1. Z = {L}
2. Find all paths from B to E.
3. For each path:
    1. Chain? If yes, are all the intermediate nodes in Z?
    2. Fork? If yes, are all the intermediate nodes in Z?
    3. Collider? If yes, are all the intermediate nodes AND their descendants NOT in Z?

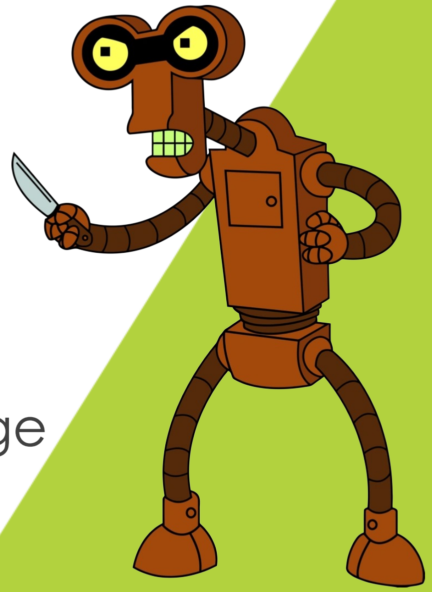
Latent nodes as partial observability



Recall: Roberto cannot directly sense whether or not there was an earthquake

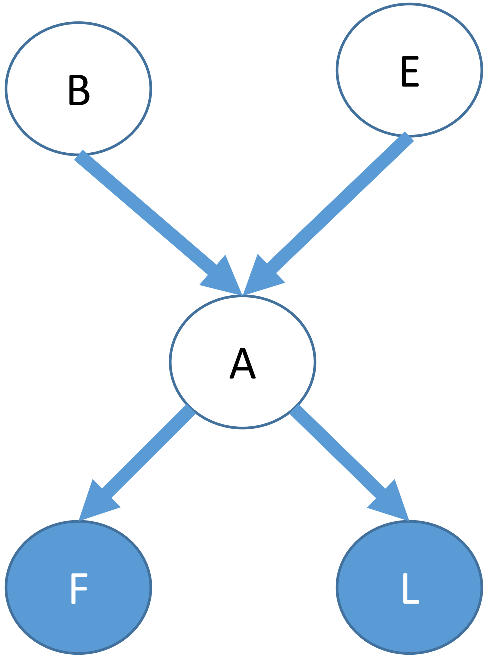
We know *post-hoc* whether there was an earthquake, but

Roberto must act without this knowledge



Implications for learning, less dire for inference/using it.

Example



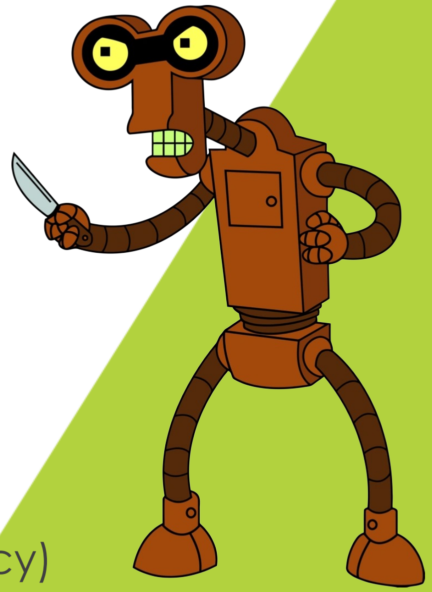
We commonly denote *unobserved* or *latent* variables using non-shaded nodes.

Roberto knows who has called.

Roberto decides whether to act menacing by sampling from

P(B | F, L)

(and possibly using other environmental Information, depending on his cost function/policy)



Learning DAG structure

Objective: learn the skeleton of a DAG

- Remember: *absence* of encodes a *structural* (path-based) independence relation
- Suppose you have many observations (i.e., rare events are not an issue)
- How might you try to learn the structure, knowing what you know about independence in the graph?

Learning DAG structure

Objective: learn t

- Remember: *ab* ation
- Suppose you h
- How might you endence in the graph?

Is U independent of V {given W}?

1. Z = W
2. Find all paths from U to W.
3. For each path:
    1. Chain? If yes, are all the intermediate nodes in Z?
    2. Fork? If yes, are all the intermediate nodes in Z?
    3. Collider? If yes, are all the intermediate nodes AND their descendants NOT in Z?

Questions to consider

What is the complexity of the search?

How can we speed things up?

What kind of background knowledge can we use?

Are there any challenges to learning the DAG you can think of?