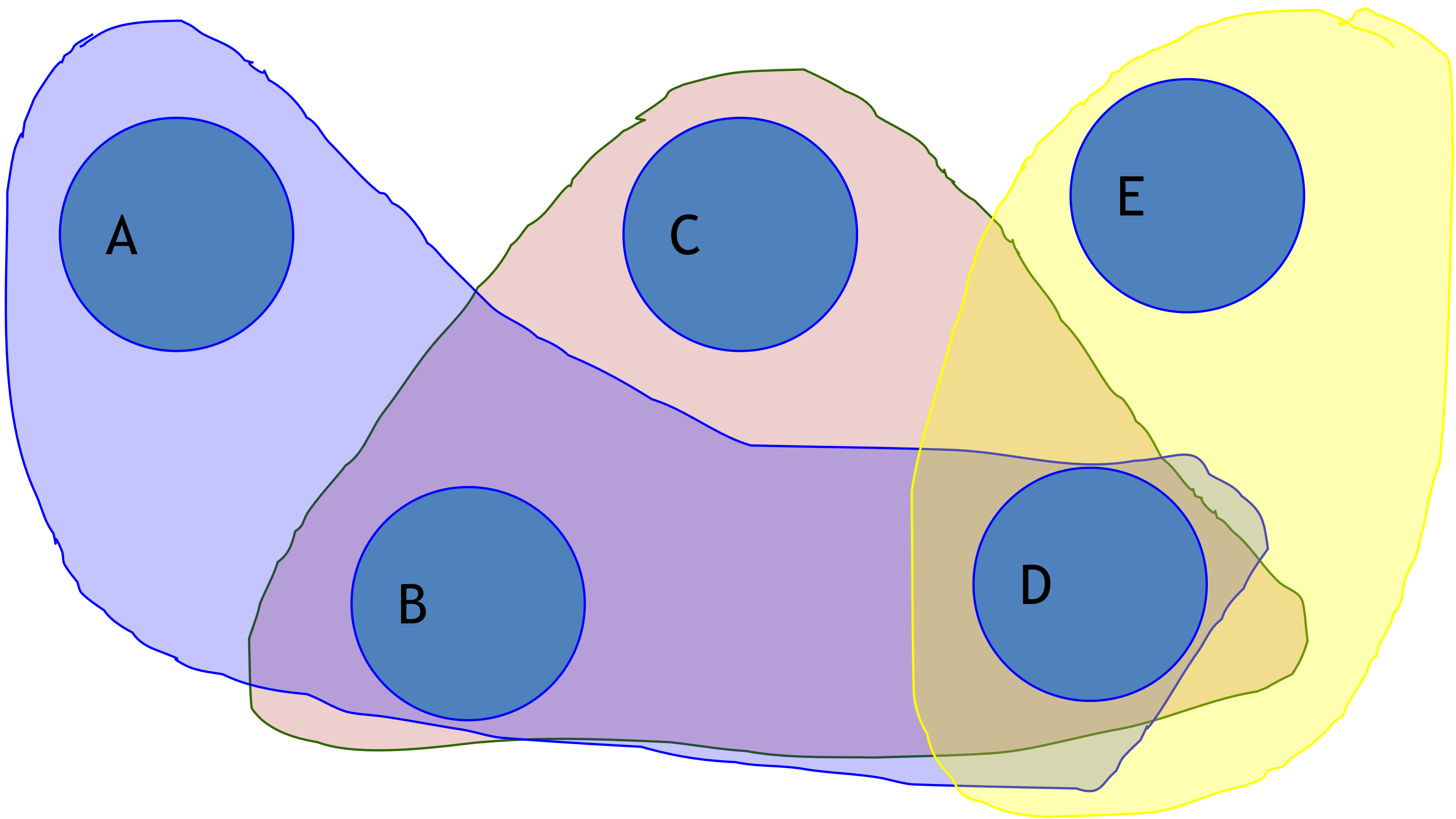
Back to … Hypergraphs & VE Complexity

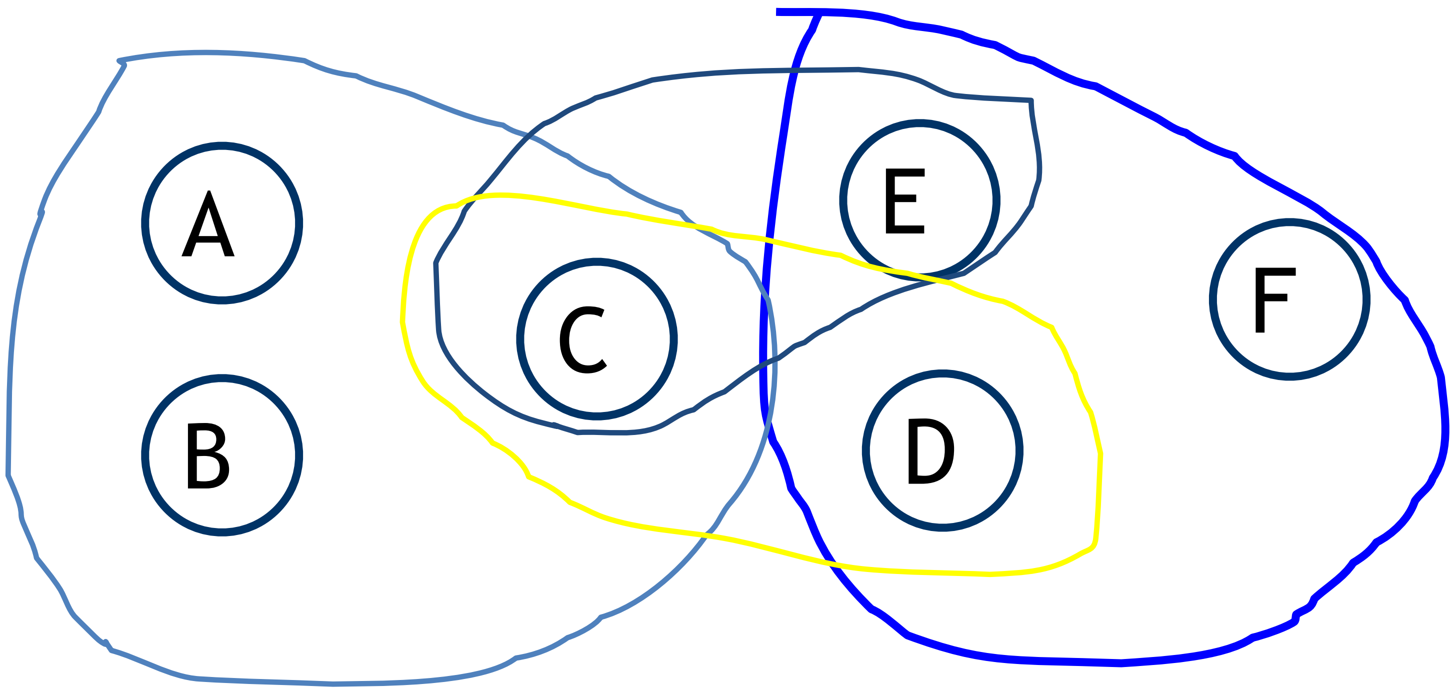
A **hypergraph** has vertices just like an ordinary graph, but instead of edges between two vertices X↔Y it contains **hyperedges**.

- A hyperedge is a set of vertices (i.e., potentially more than one)

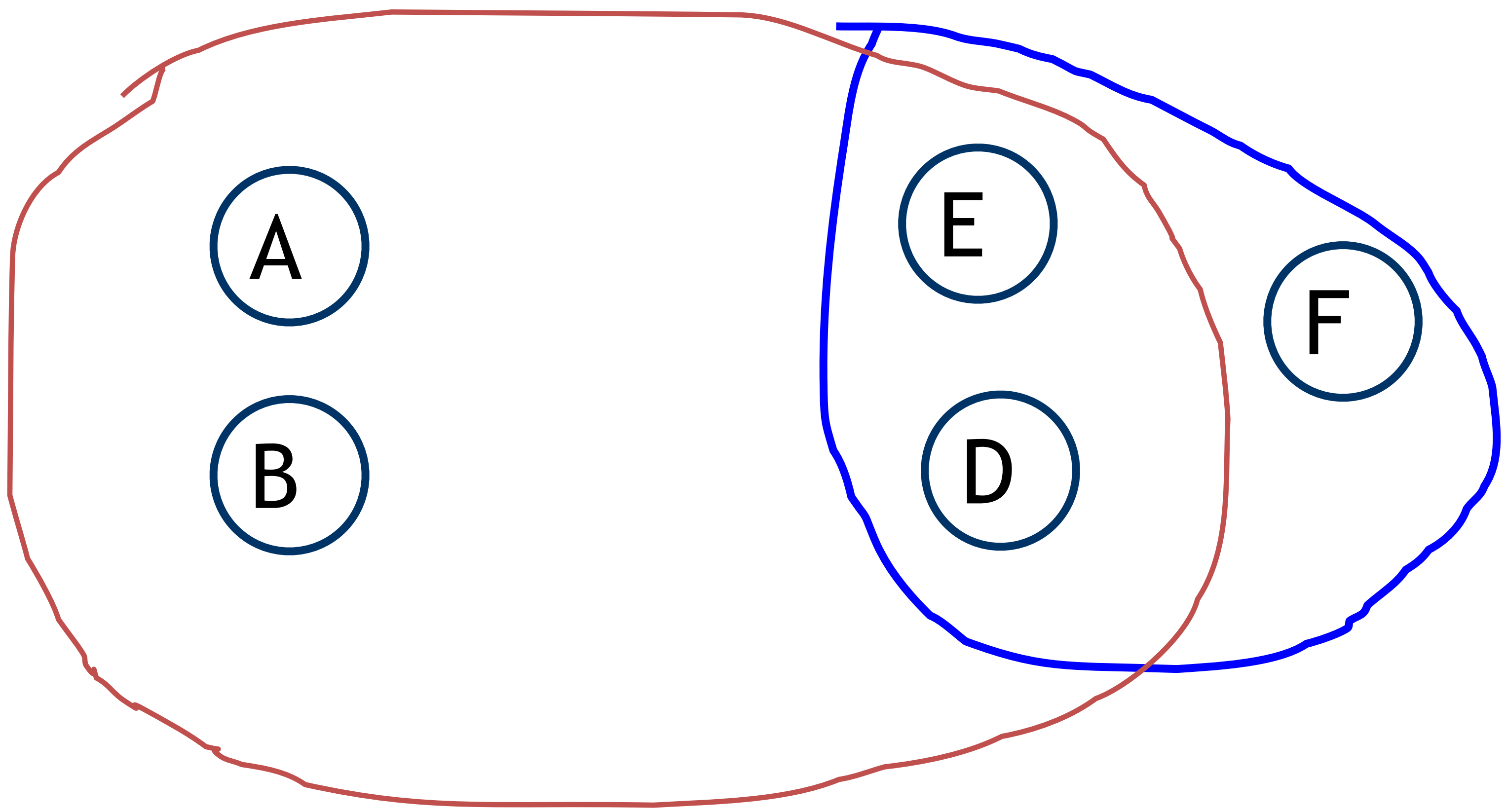


{A,B,D}
{B,C,D}
{E,D}

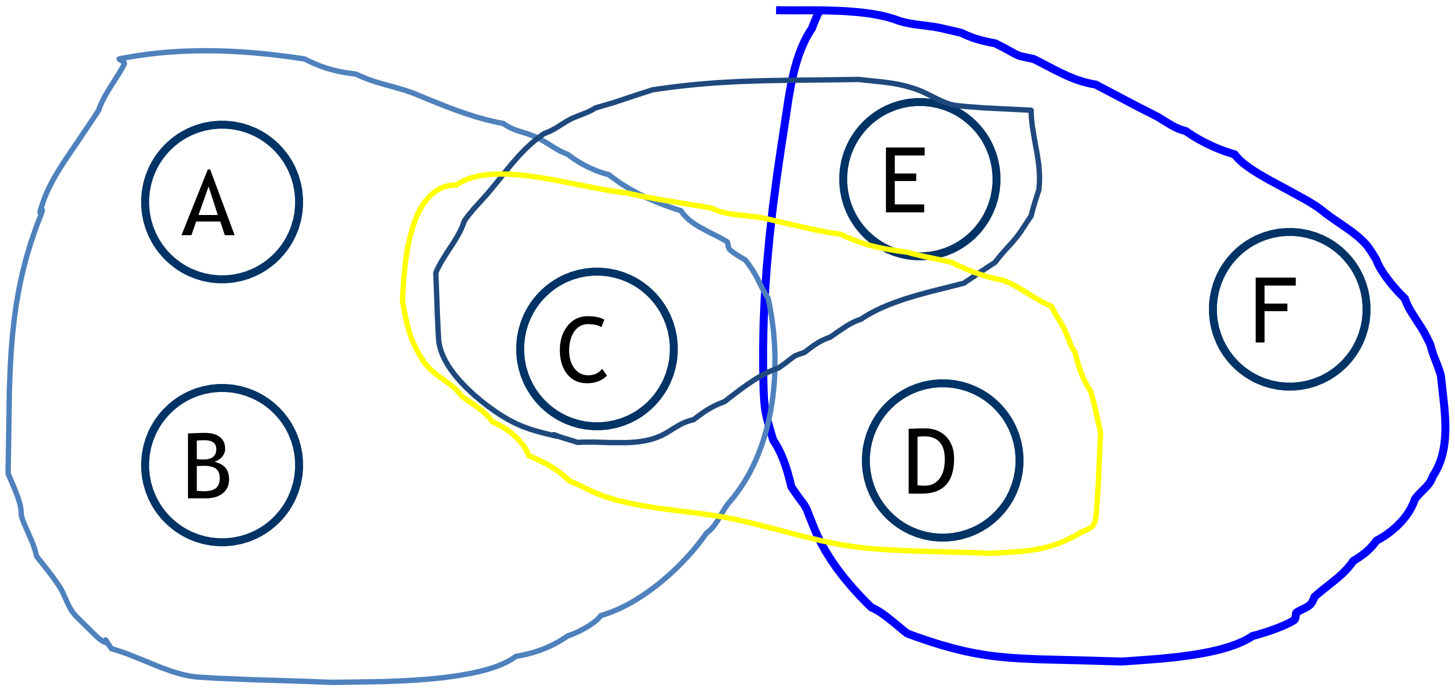
Hypergraphs and Complexity of VE



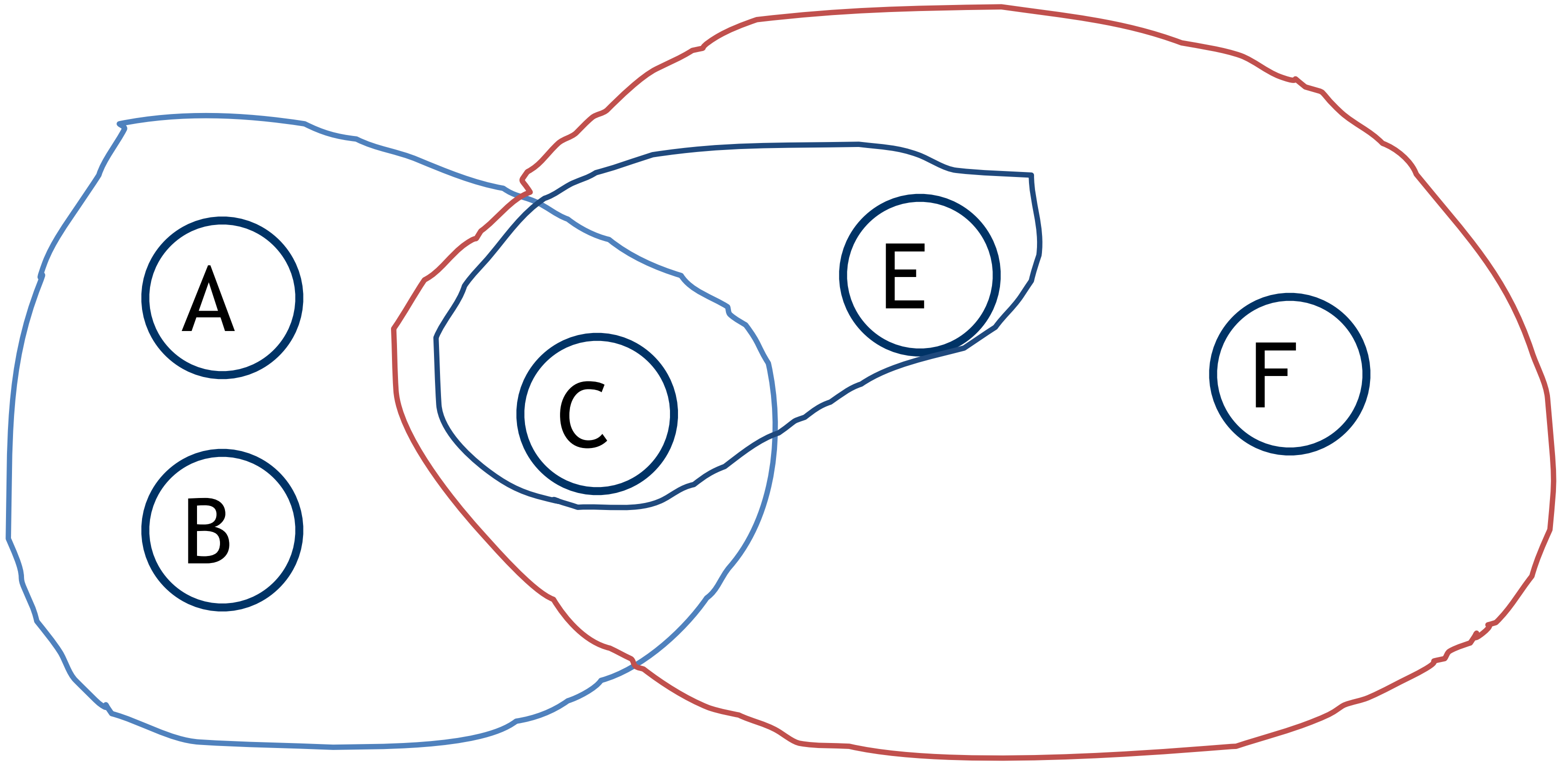
- Eliminate C



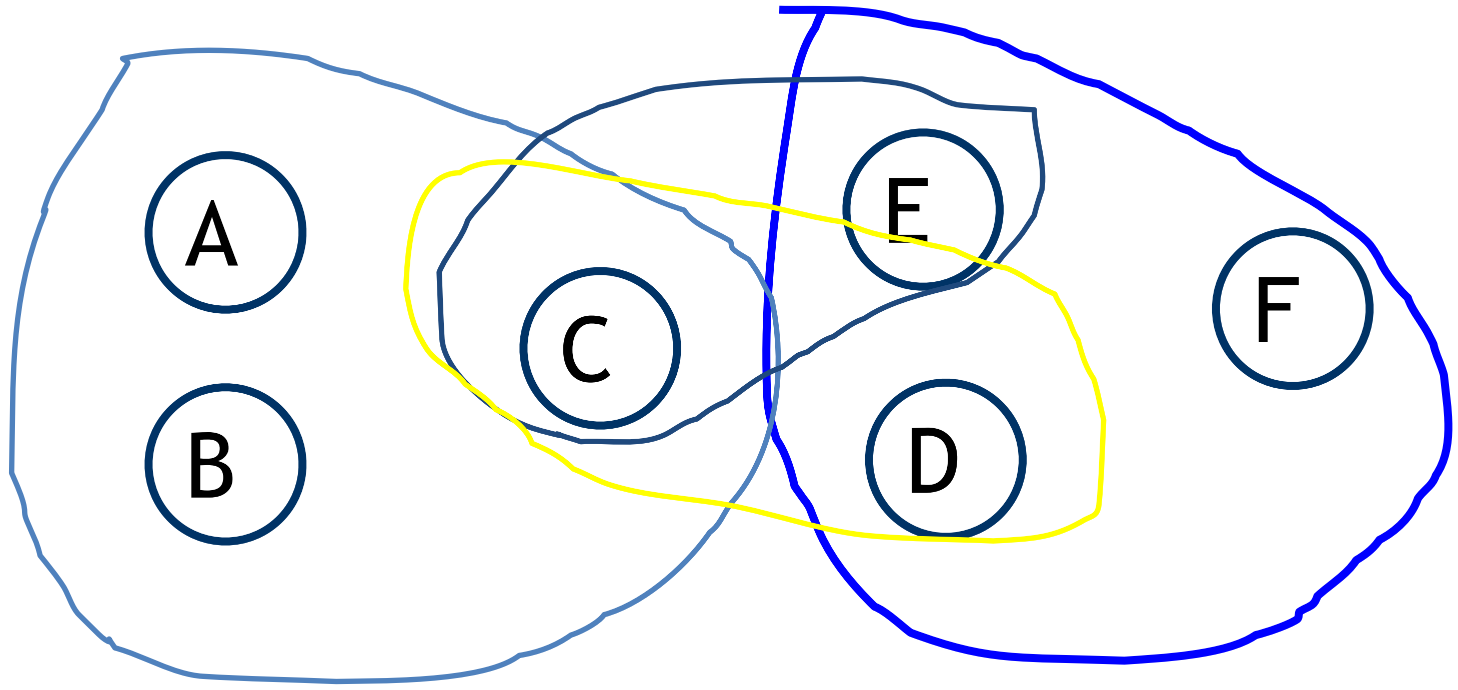
Hypergraphs and Complexity of VE



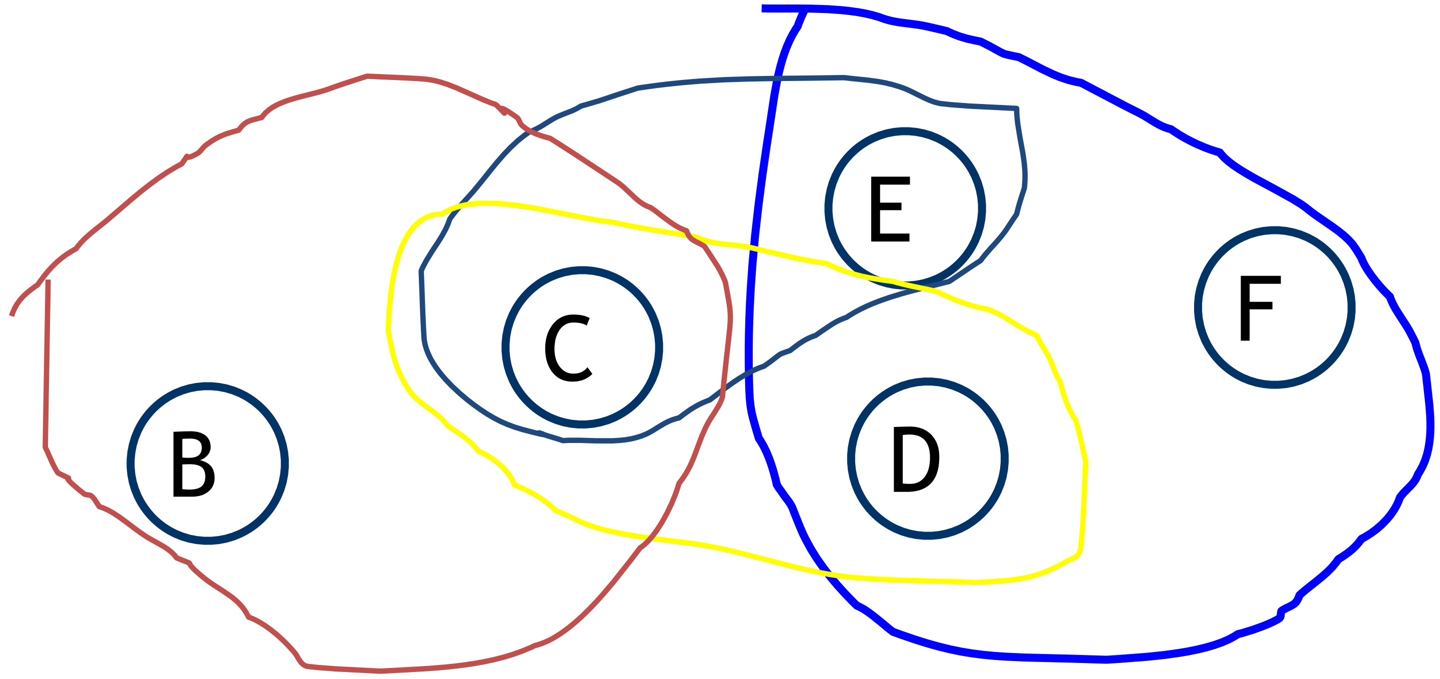
- Eliminate D



Hypergraphs and Complexity of VE



- Eliminate A



Elimination Width

- Given an ordering π of the variables and an initial hypergraph $\mathscr{H}$ eliminating these variables yields a sequence of hypergraphs

  $\mathscr{H} = H_0, H_1, H_2, \ldots, H_n$

  where H_n contains only one vertex (the query variable).

- The elimination width π is the maximum size (number of variables) of any hyperedge in any of the hypergraphs $H_0, H_1, \ldots, H_n$.
- The induced width of the previous example was 4 ({A,B,E,D} in H_1 and H_2).

Elimination Width

- If the elimination width of an ordering π is k, then the complexity of VE using that ordering is $2^{O(k)}$
- Elimination width k means that at some stage in the elimination process a factor involving k variables was generated.
- That factor will require $2^{O(k)}$ space to store
  - space complexity of VE is $2^{O(k)}$
- And it will require $2^{O(k)}$ operations to process (either to compute in the first place, or when it is being processed to eliminate one of its variables).
  - Time complexity of VE is $2^{O(k)}$
- NOTE, that k is the elimination width of this particular ordering.

Tree Width

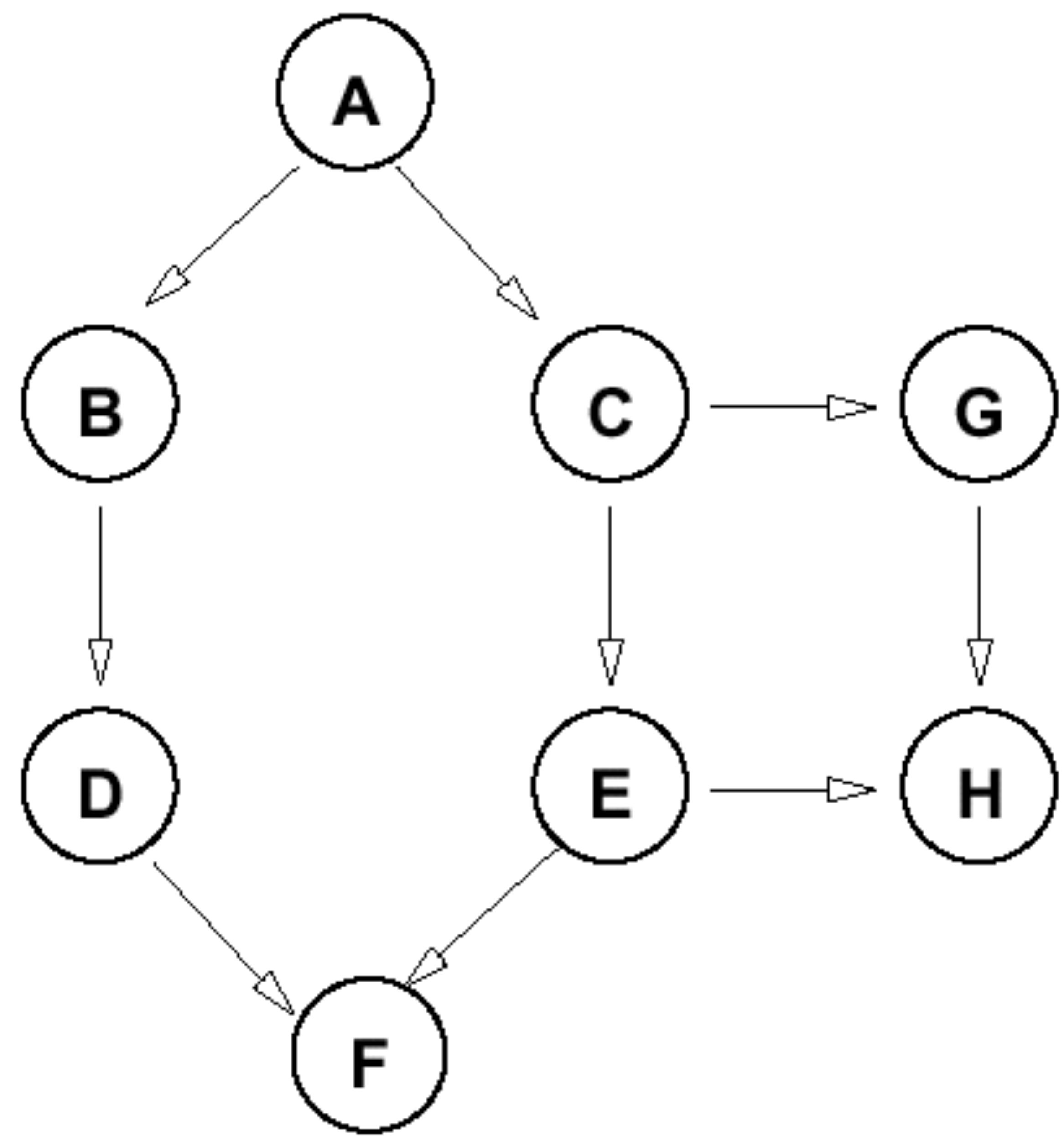
- Given a hypergraph $\mathscr{H}$ with vertices $\{X_1,X_2,\ldots,X_n\}$ the tree width (ω) of $\mathscr{H}$ is the MINIMUM elimination width of *any of the n!* different orderings of the X_i minus *1*.

- Thus VE has best case complexity of $2^{O(\omega)}$ where ω is the tree width of the initial Bayes Net.

- In the worst case, the tree width is equal to the number of variables (minus 1)

Different Orderings = Different Elimination Widths

Suppose query variable is D. Consider different orderings for this network



E,C,A,B,G,H,F :

- Bad

A,F,H,G,B,C,E:

- Good

Tree Width

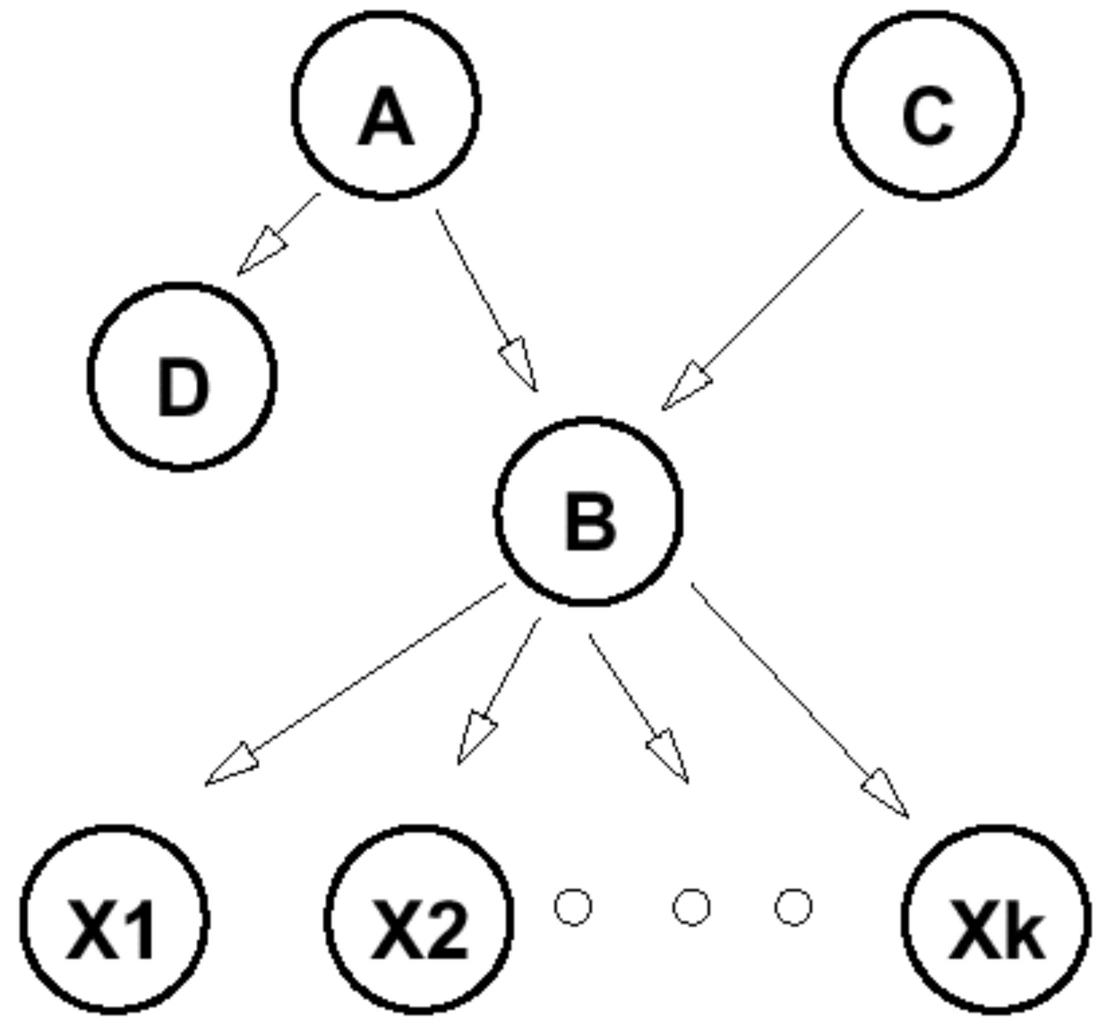
- Exponential in the tree width is the best that VE can do.
  - But, finding an ordering that has elimination width equal to tree width is NP-Hard.
    - so in practice there is no point in trying to speed up VE by finding the best possible elimination ordering.
  - Instead, heuristics are used to find orderings with good (low) elimination widths.
  - In practice, this can be very successful. Elimination widths can often be relatively small, 8-10 even when the network has 1000s of variables.
    - Thus VE can be *much* more efficient than simply summing the probability of all possible events (which is exponential in the number of variables).
    - Sometimes, however, the tree width is equal to the number of variables (minus 1).

Finding Good Orderings

- A *polytree* is a singly connected Bayes Net: in particular there is only one path between any two nodes.
- A node can have multiple parents, but we have no cycles.
- Good orderings are easy to find for polytrees
  - At each stage eliminate *a singly connected node*.
  - Because we have a polytree we are assured that a singly connected node will exist at each elimination stage.
  - The size of the factors in the tree never increase.

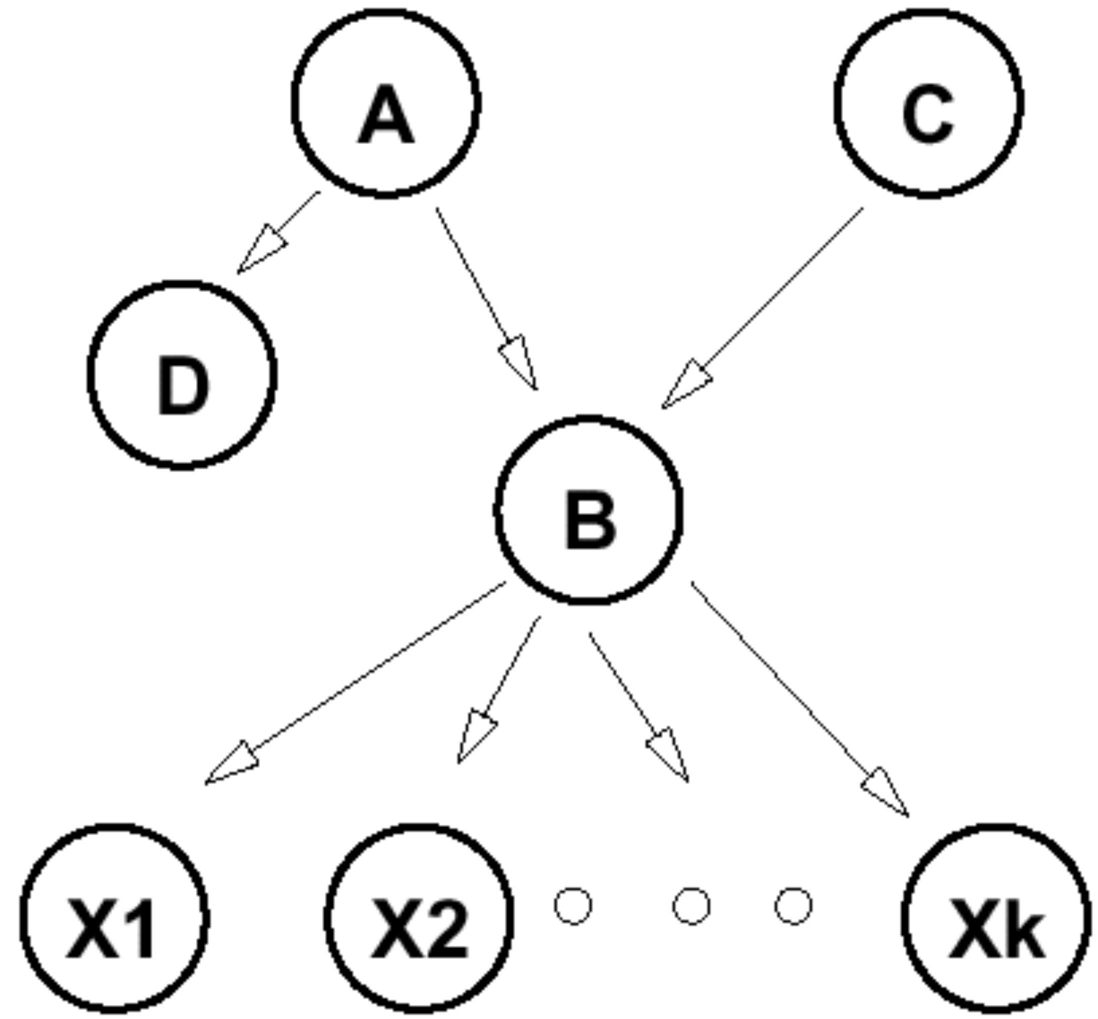
Elimination Ordering: Polytrees

- Eliminating singly connected nodes allows VE to run in linear time
  - e.g., in this network, eliminate D, A, C, X1,…; or eliminate X1,… Xk, D, A C; or mix it up.
  - result: no factor ever larger than original CPTs
  - eliminating B before these gives factors that include all of A,C, X1,… Xk!!!

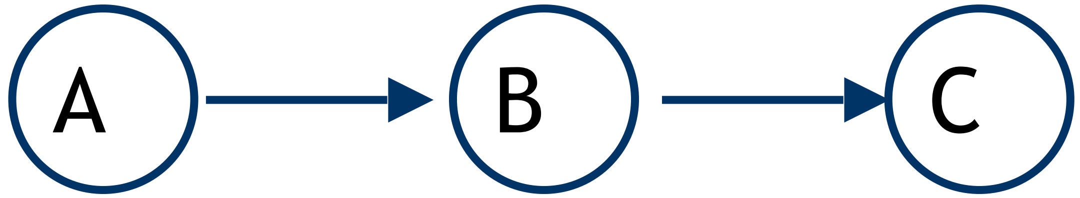


Min Fill Heuristic

- A fairly effective heuristic is to always eliminate next the variable that creates the smallest size factor.
- This is called the min-fill heuristic.
  - –B creates a factor of size k+2
  - –A creates a factor of size 2
  - –D creates a factor of size 1
- This heuristic always solves polytrees in linear time.



Relevance

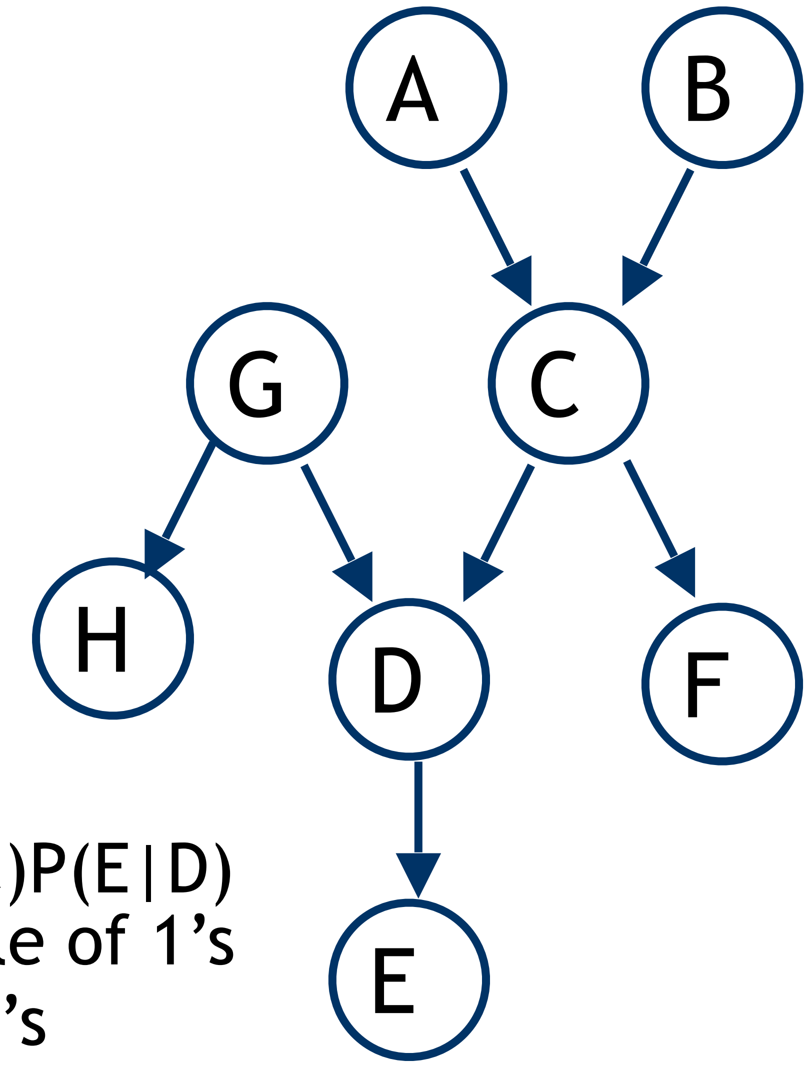


- Certain variables have no impact on the query. For example, in network ABC, computing P(A) with no evidence requires elimination of B and C.
  - But when you sum out these variables, you compute a trivial factor (that always evaluates to 1); for example:
  - Eliminating C: $f_4(B) = \Sigma_C\, f_3(B,C) = \Sigma_C\, P(C|B)$
  - This is 1 for any value of B (e.g., $P(c|b) + P(\sim c|b) = 1$)
- No need to think about B or C for this query

Relevance

- Can restrict attention to *relevant* variables. Given query **q**, evidence **E**:
  - **q** itself is relevant
  - if any node **Z** is relevant, its parents are relevant
  - if e∈**E** is a descendent of a relevant node, then E is relevant
- We can restrict our attention to the *sub-network comprising only relevant variables* when evaluating a query Q

Relevance: Examples



- Query: P(F)
  - relevant: F, C, B, A
- Query: P(F|E)
  - relevant: F, C, B, A
  - also: E, hence D, G
  - *intuitively, we need to compute P(C|E) to compute P(F|E)*
- Query: P(F|H)
  - relevant: F, C, B, A

  $P(A)P(B)P(C|A,B)P(F|C)\ P(G)P(h|G)P(D|G,C)P(E|D)$

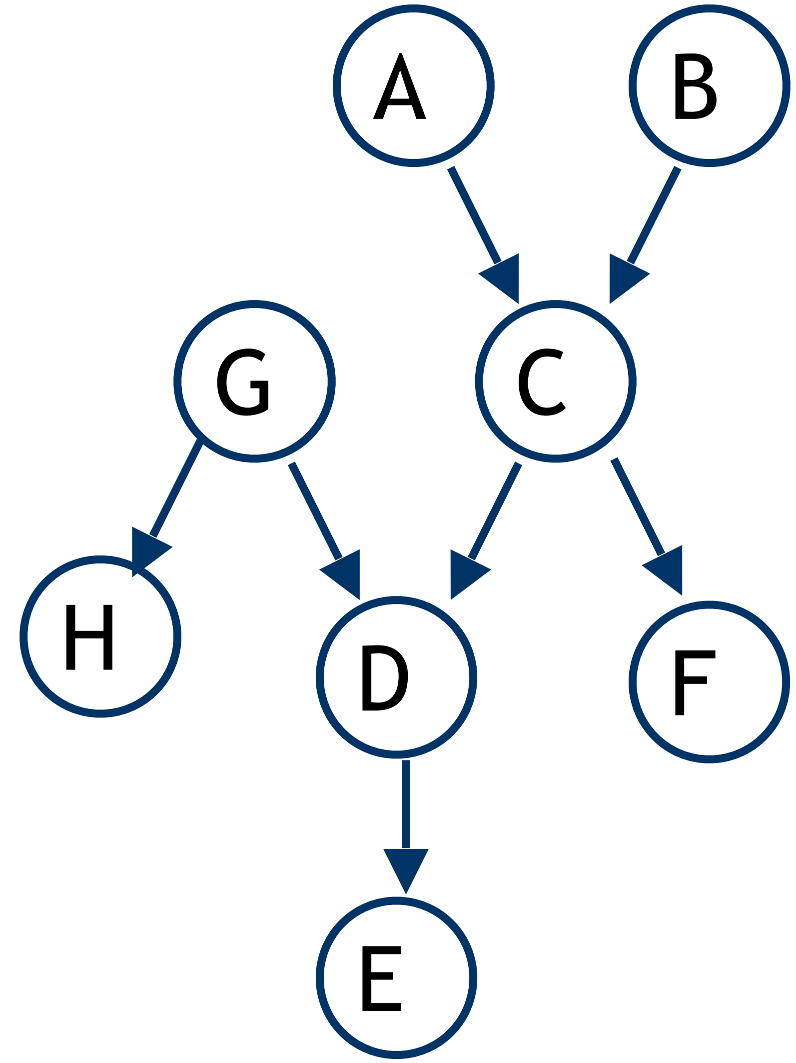
  $= \ldots\ P(G)P(h|G)P(D|G,C)\ \sum_E P(E|D)$ = a table of 1's

  $= \ldots\ P(G)P(h|G)\ \sum_D P(D|G,C)$ = a table of 1's

  $= [P(A)P(B)P(C|A,B)P(F|C)]\ [\sum_G P(G)P(h|G)]$

  $[\sum_G P(G)P(h|G)] \neq 1$ but irrelevant once we normalize, multiplies each value of F equally

Relevance: Examples



Query: P(F|E,C)

- algorithm says all variables except H are relevant; but really none except C, F (since C cuts of all influence of others)
- algorithm is overestimating relevant set

Independence in a Bayes Net

- Another piece of information we can obtain from a Bayes net is the "structure" of relationships in the domain.
- The structure of the BN means: every X_i is *conditionally independent of all of its non-descendants given it parents*:

$$P(X_i \mid S \cup Par(X_i)) = P(X_i \mid Par(X_i))$$

for any subset $S \subseteq NonDescendents(X_i)$

More generally ….

Conditional independencies can be useful in computation, explanation, etc. related to a BN

How do we determine if two variables X, Y are independent given a set of variables E?

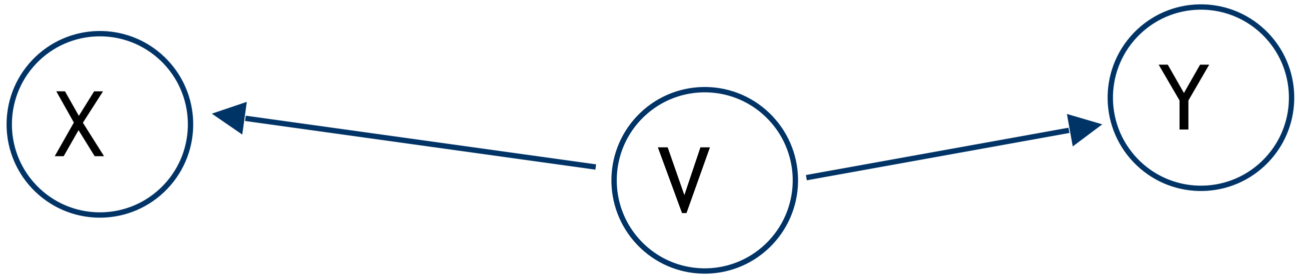
- We can use a (simple) graphical property called **D-separation**

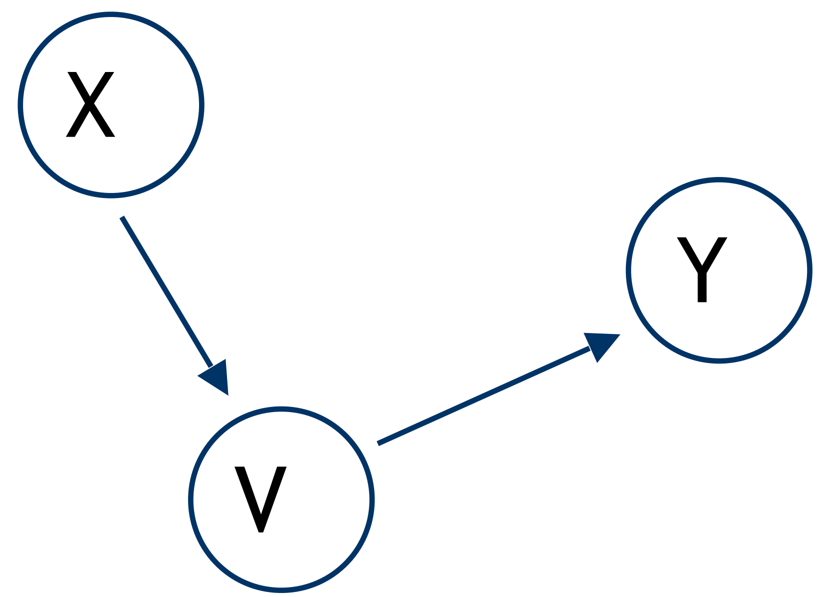
D-separation: A set of variables E **d-separates** X and Y if it **blocks** every undirected path in the BN between X and Y (we'll define Blocks next.)

X and Y are conditionally independent given evidence E if E **d- separates** X and Y

- thus BN gives us an easy way to tell if two variables are independent (set E = {}) or conditionally independent given E.

What does it mean to be blocked?

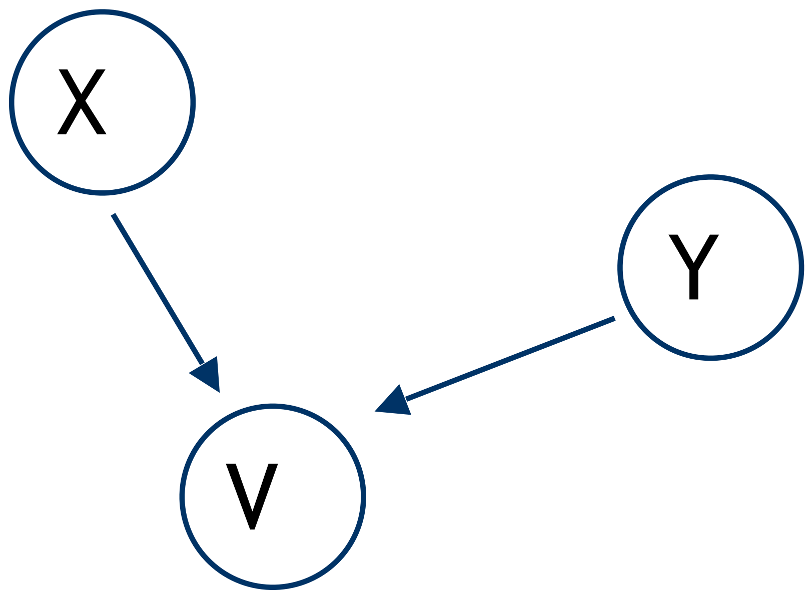




There exists a variable *V* on the path such that
- it **is** in the evidence set *E*
- the arcs putting *V* in the path are "tail-to-tail"

Or, there exists a variable *V* on the path such that
- it **is** in the evidence set *E*
- the arcs putting *V* in the path are "tail-to-head"

What does it mean to be blocked?



If the variable V is on the path such that the arcs putting V on the path are "head-to-head", the variables are still blocked .... so long as:

V is NOT in the evidence set E

neither are any of its descendants

Blocking: Graphical View

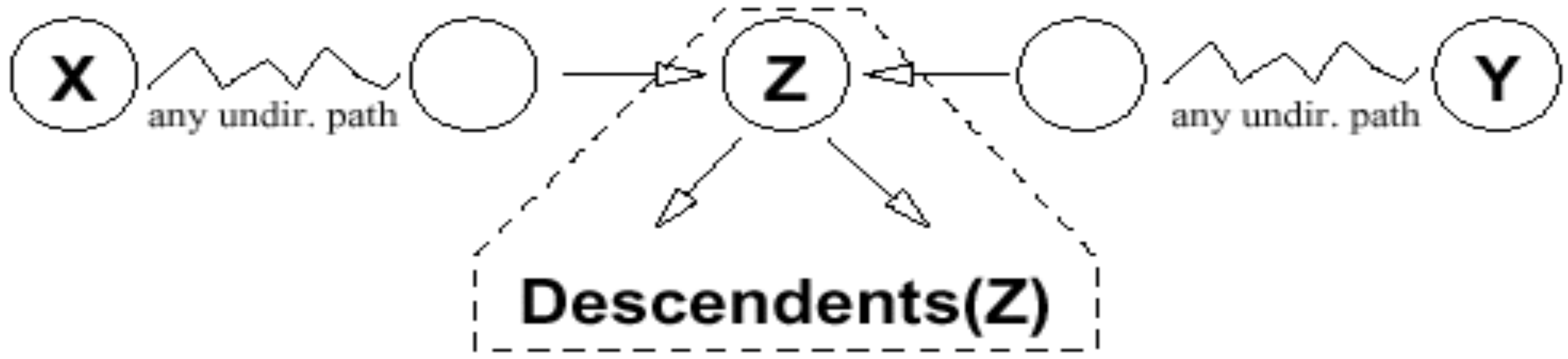
(1)

If Z in evidence, the path between X and Y blocked

(2)

If Z in evidence, the path between X and Y blocked

(3)

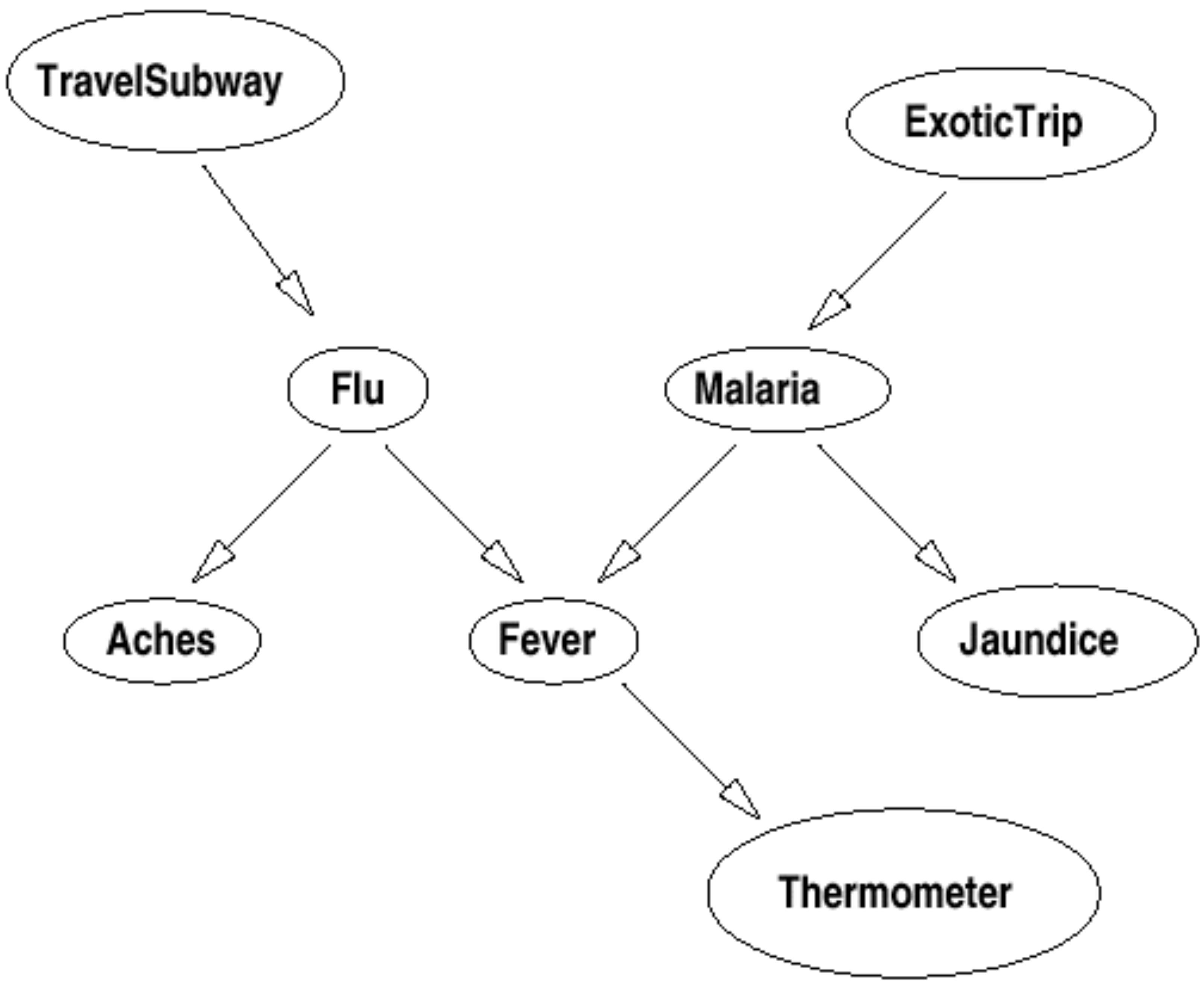


If Z is ***not*** in evidence and ***no*** descendent of Z is in evidence, then the path between X and Y is blocked

D-separation implies conditional independence

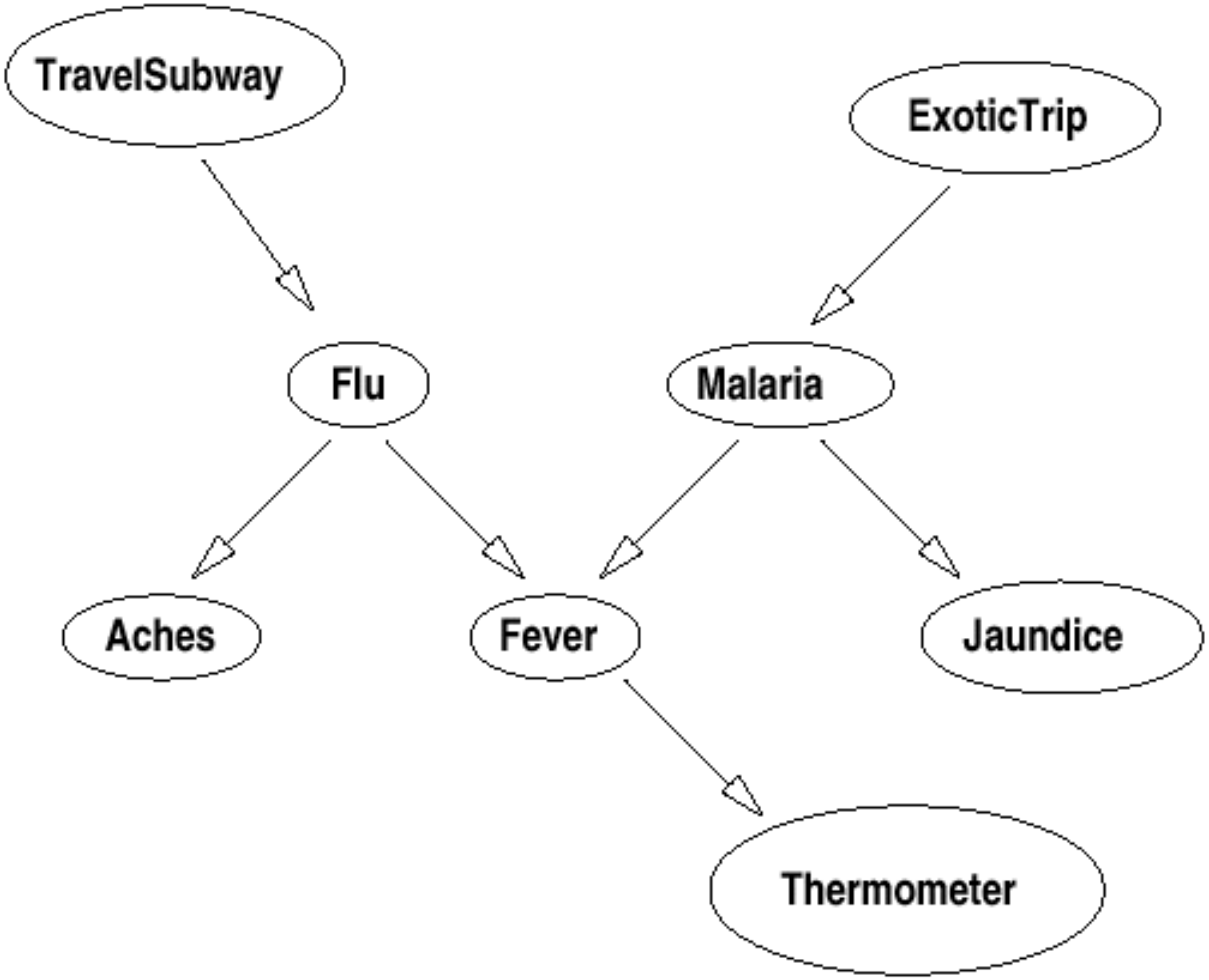
Theorem [Verma & Pearl, 1998]: If a set of evidence variables E d-separates X and Z in a Bayesian network's graph, then X is independent of Z given E.

D-Separation: Intuitions



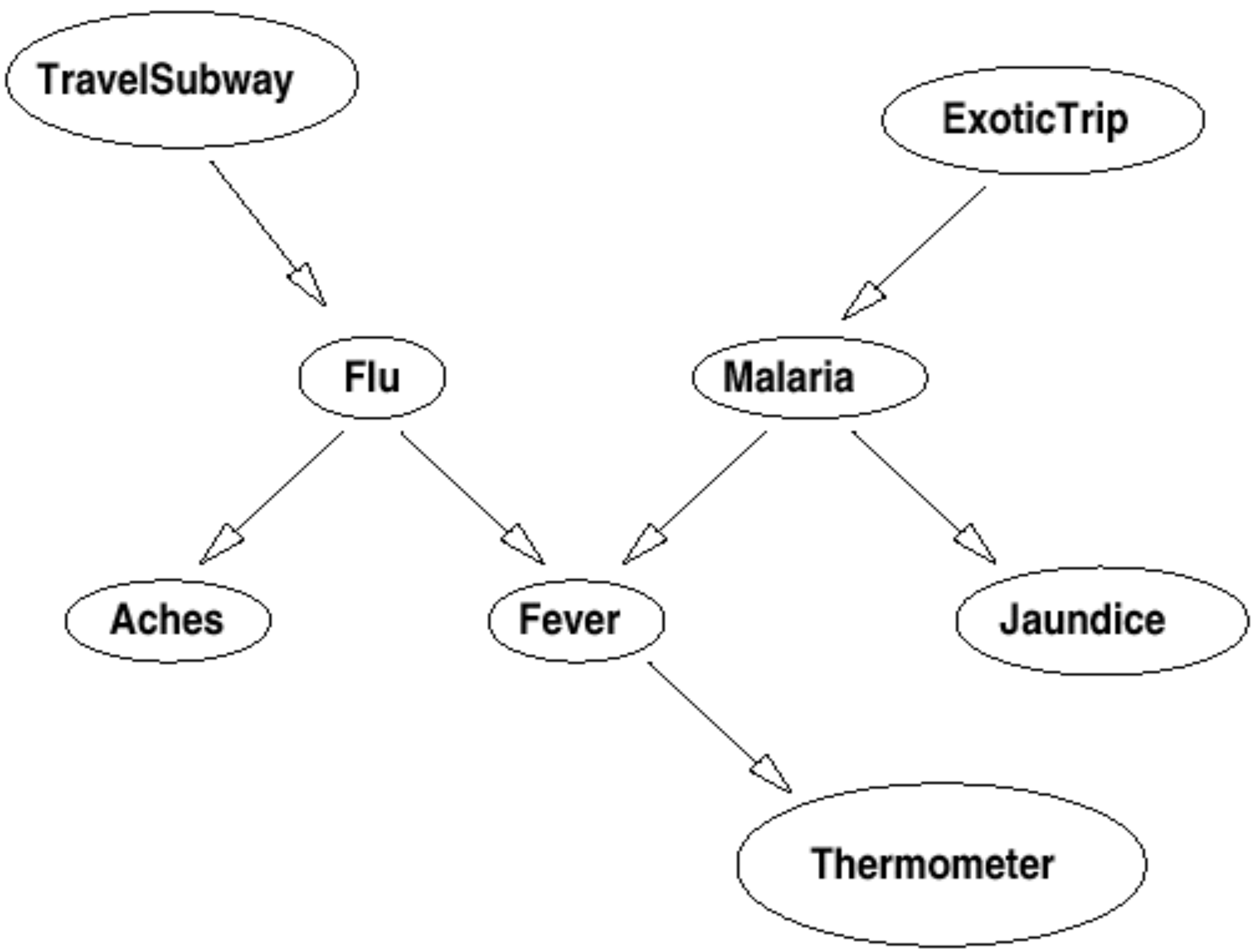
Subway and Therm are dependent; but are independent given Flu (since Flu blocks the only path)

D-Separation: Intuitions



Aches and Fever are dependent; but are independent given Flu (since Flu blocks the only path). Similarly for Aches and Therm (dependent, but indep. given Flu).

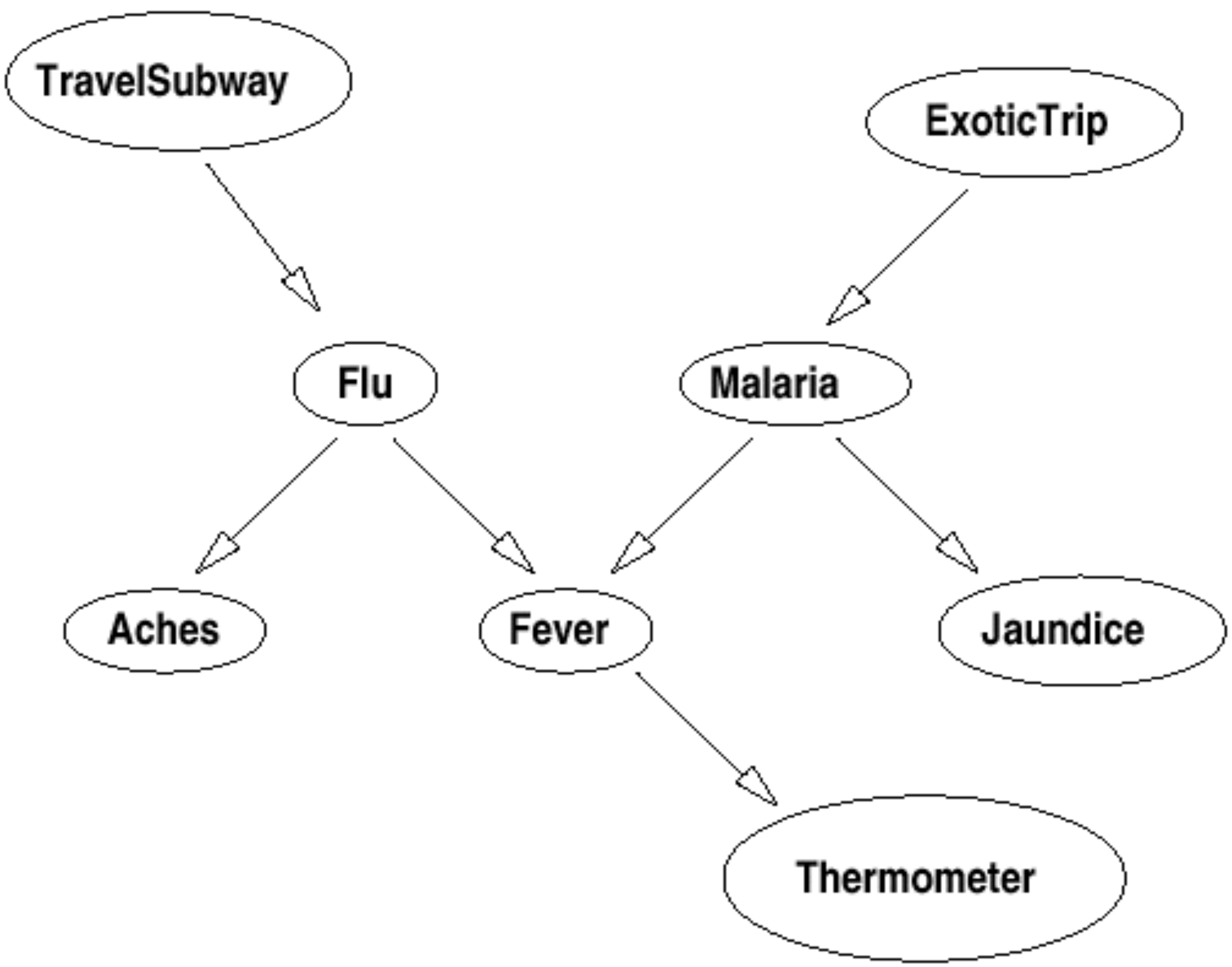
D-Separation: Intuitions



Flu and Mal are indep. (given no evidence): Fever blocks the path, since it is not in evidence, nor is its descendant Therm.

Flu and Mal are dependent given Fever (or given Therm): nothing blocks path now.

D-Separation: Intuitions



Subway, ExoticTrip are indep.;

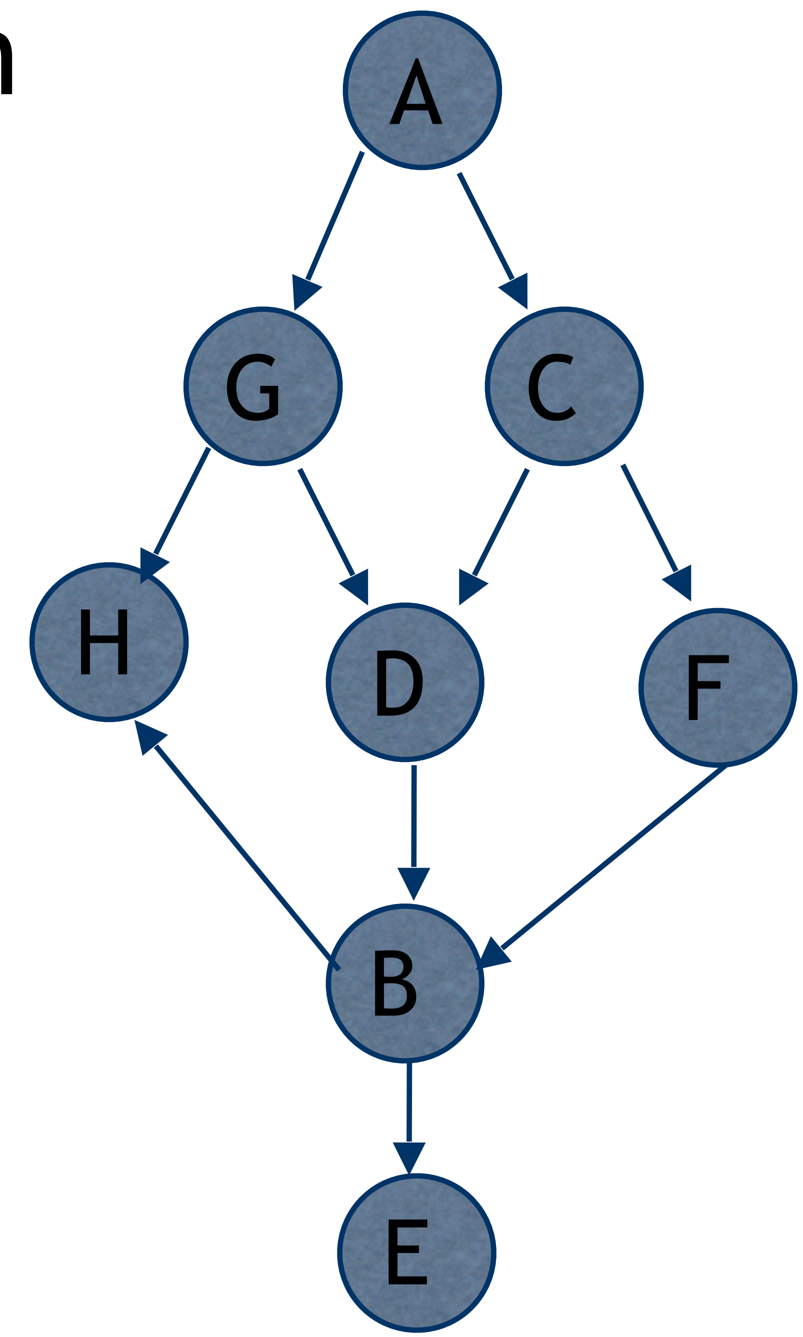
they are dependent given Therm;

they are indep. given Therm and Malaria. This for exactly the same reasons for Flu/Mal above.

D-Separation Example

In the following network determine if A and E are independent given the evidence.

1. A and E given no evidence?
2. A and E given {C}?
3. A and E given {G,C}?
4. A and E given {G,C,H}?
5. A and E given {G,F}?
6. A and E given {F,D}?
7. A and E given {F,D,H}?
8. A and E given {B}?
9. A and E given {H,B}?
10. A and E given {G,C,D,H,D,F,B}?



D-Separation Example

In the following network determine if A and E are independent given the evidence.

1. A and E given no evidence? N
2. A and E given {C}? N
3. A and E given {G,C}? Y
4. A and E given {G,C,H}? Y
5. A and E given {G,F}? N
6. A and E given {F,D}? Y
7. A and E given {F,D,H}? N
8. A and E given {B}? Y
9. A and E given {H,B}? Y
10. A and E given {G,C,D,H,D,F,B}? Y