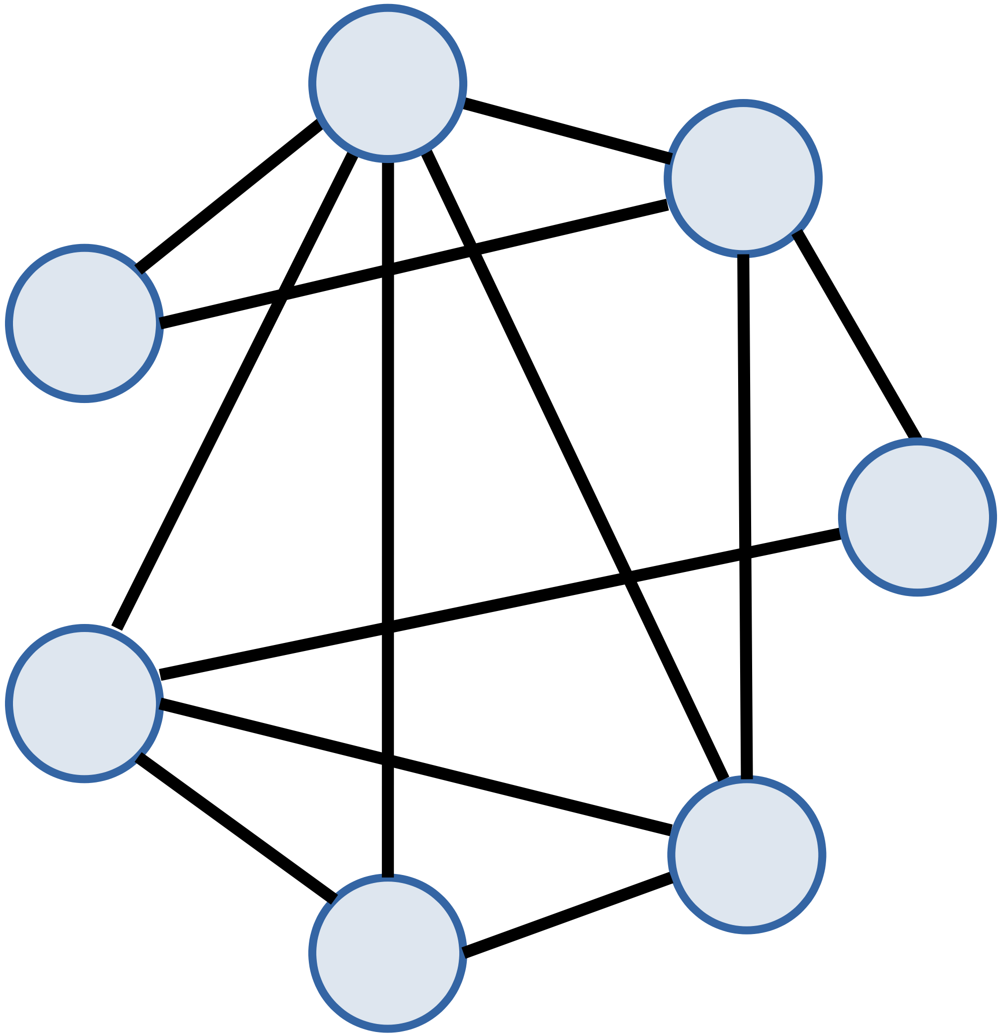
Causal Discovery Algorithms

Agenda

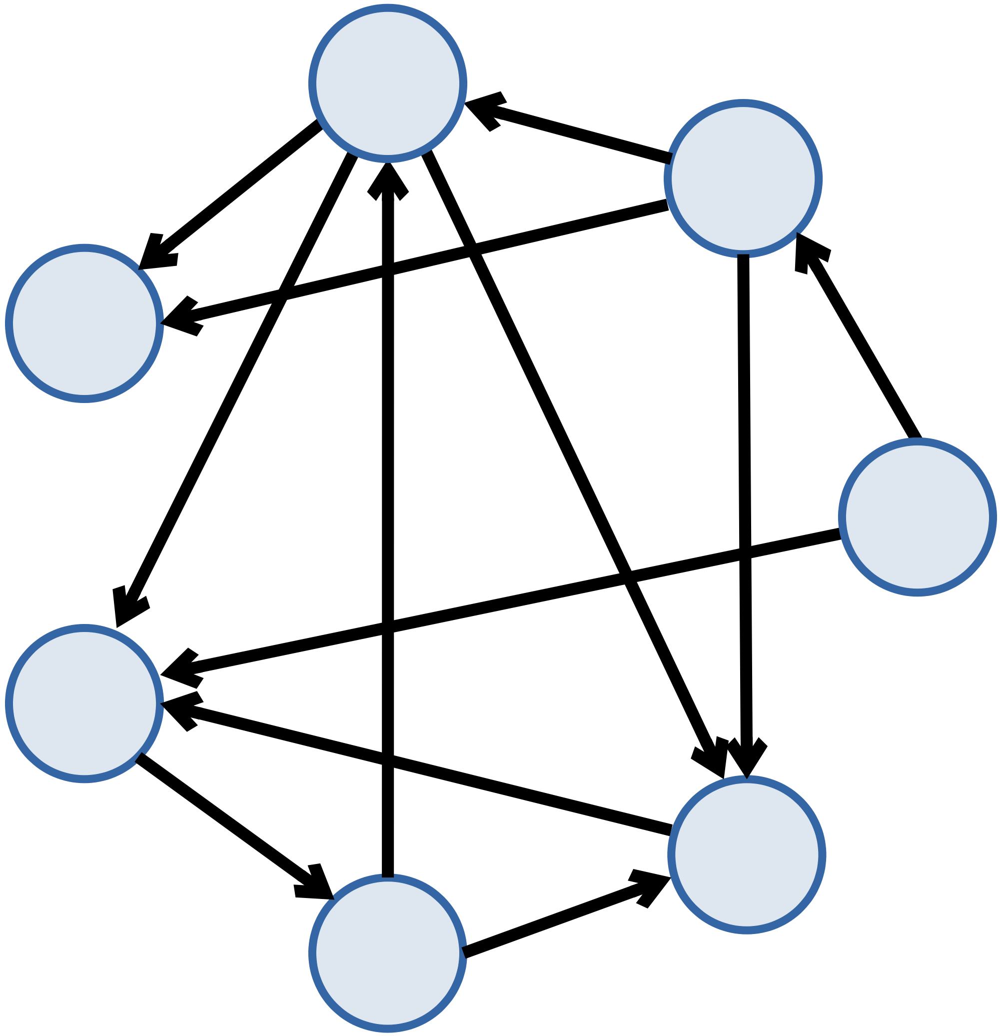
- Discovery vs inference
- Causal discovery algorithms
  - PC
  - GES
  - LiNGAM
  - NOTEARS
  - Deep Learning
  - LLMs
- Code and examples

Discovery
vs
Inference

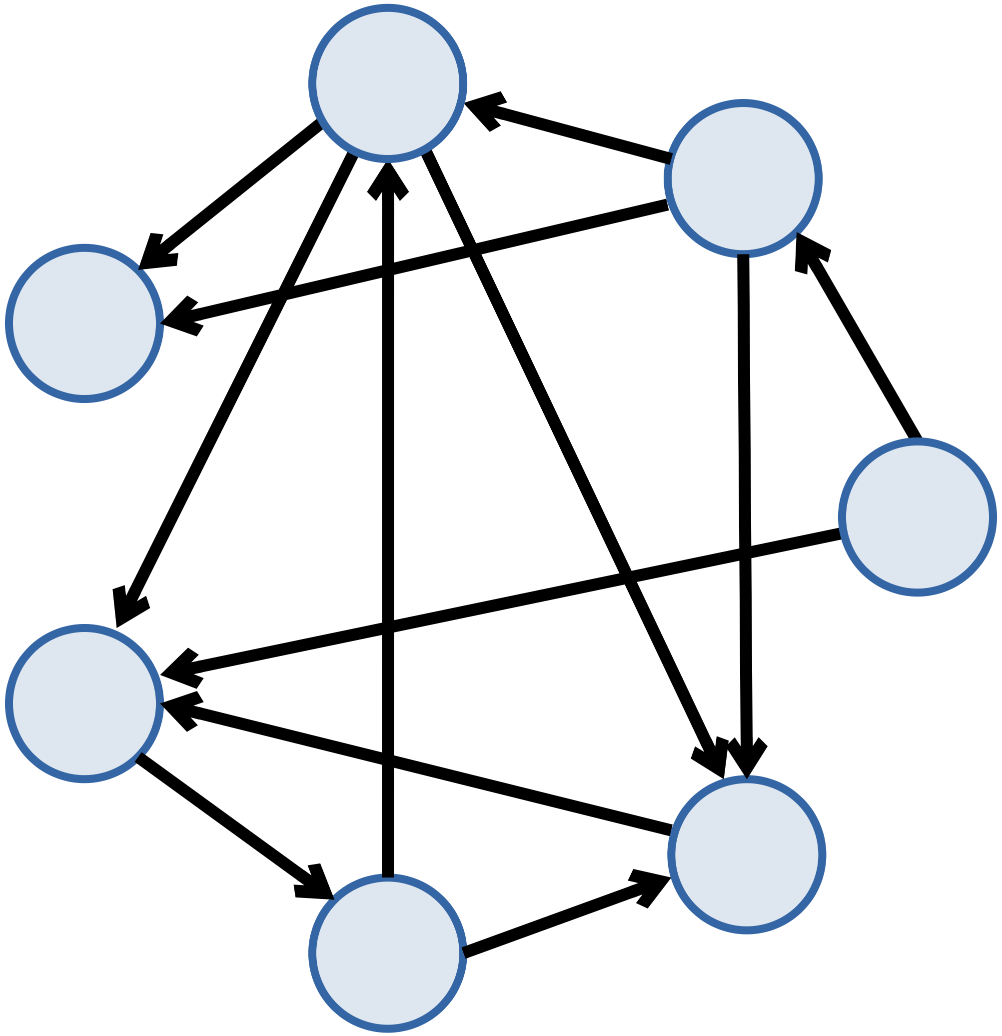
Graph



Directed graph



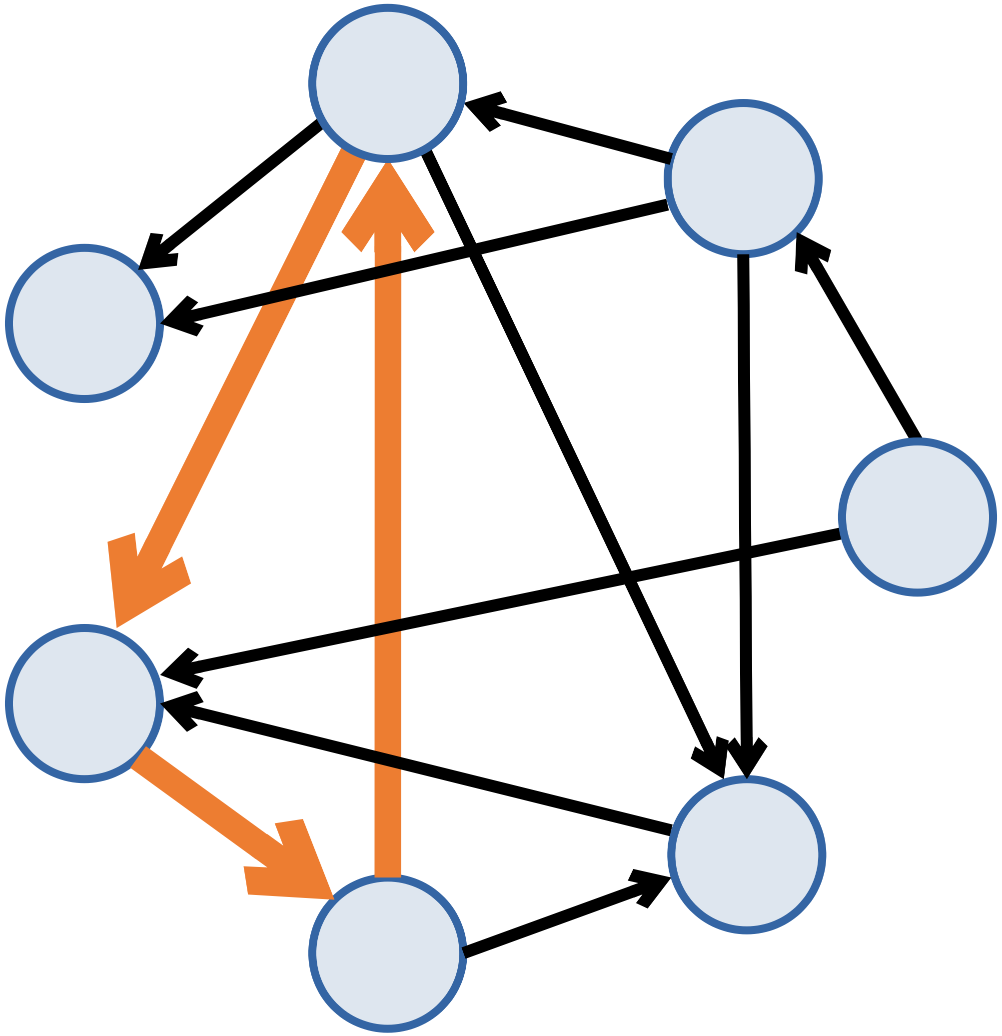
Directed graph



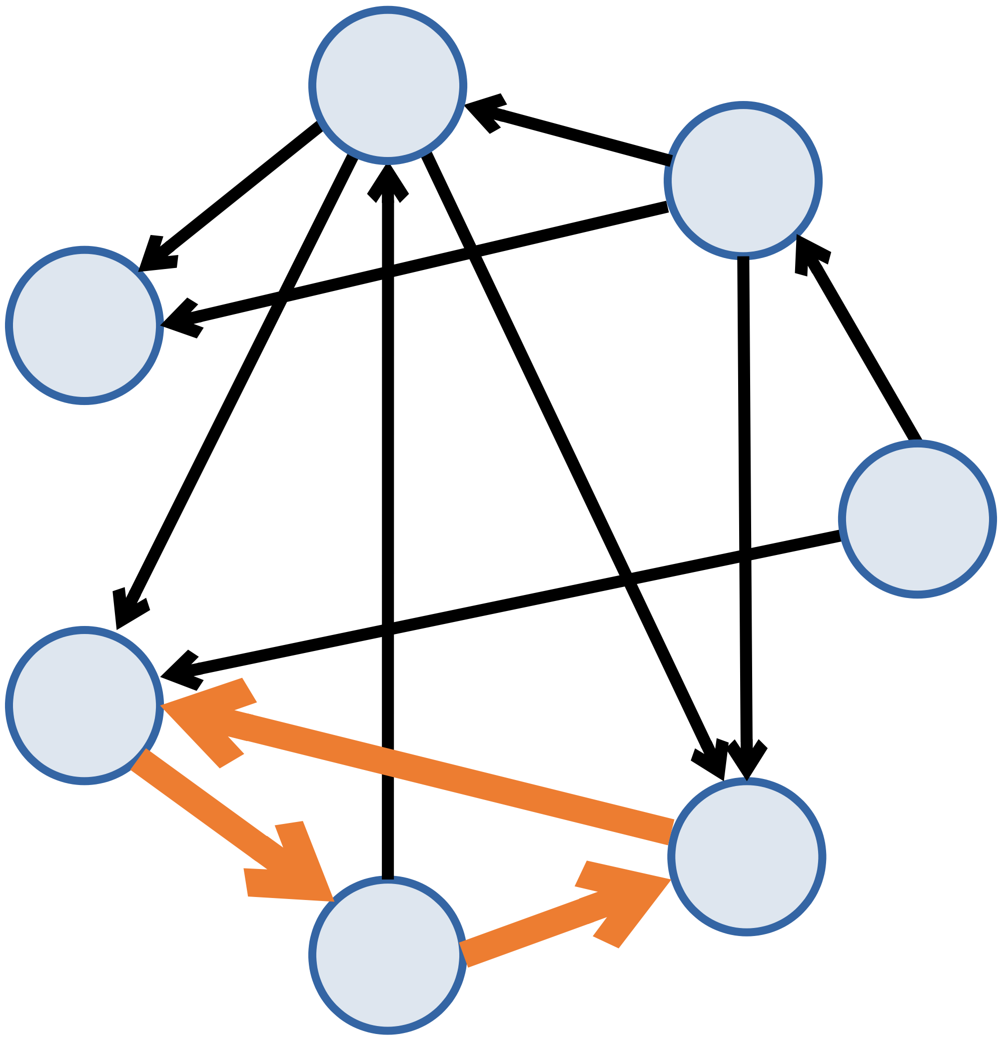
A directed graph (V,E) is:

- a set V (vertices, or nodes)
- and a set $E \subset V \times V$ (edges).

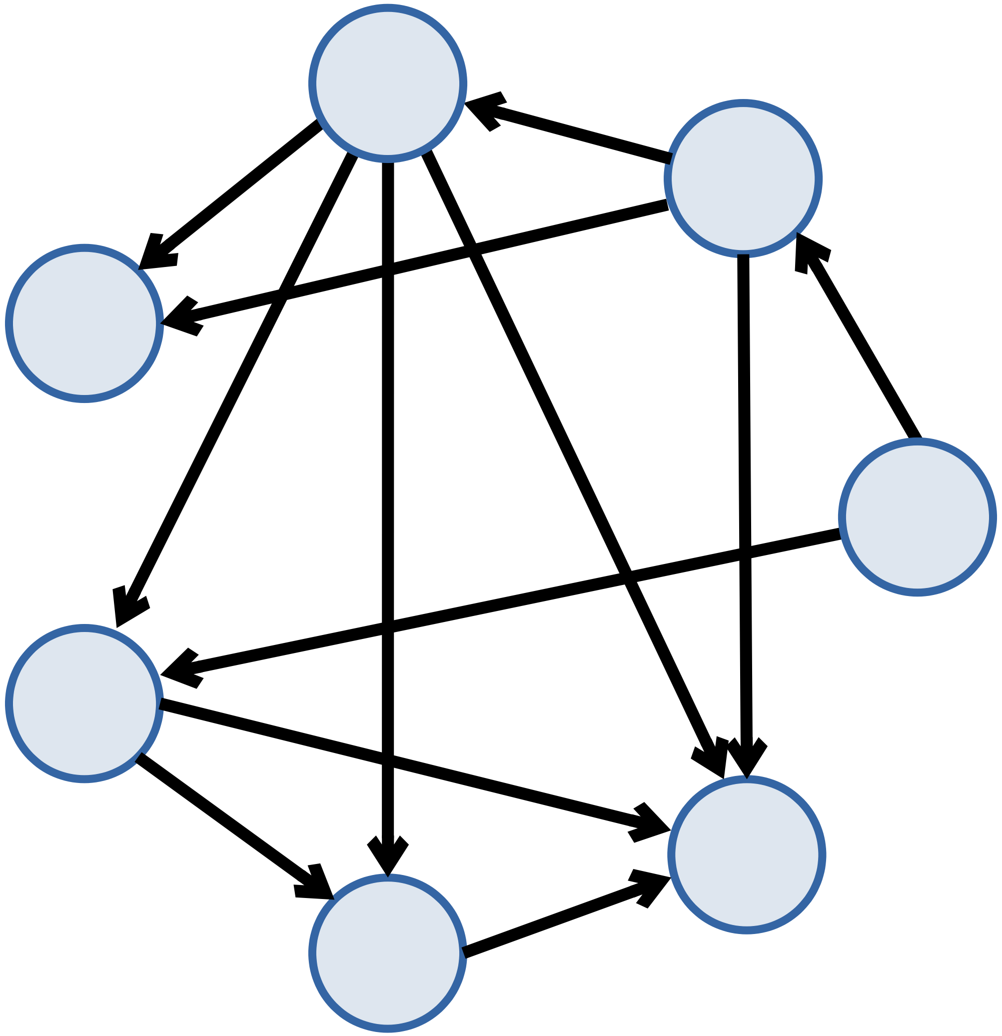
Directed graph



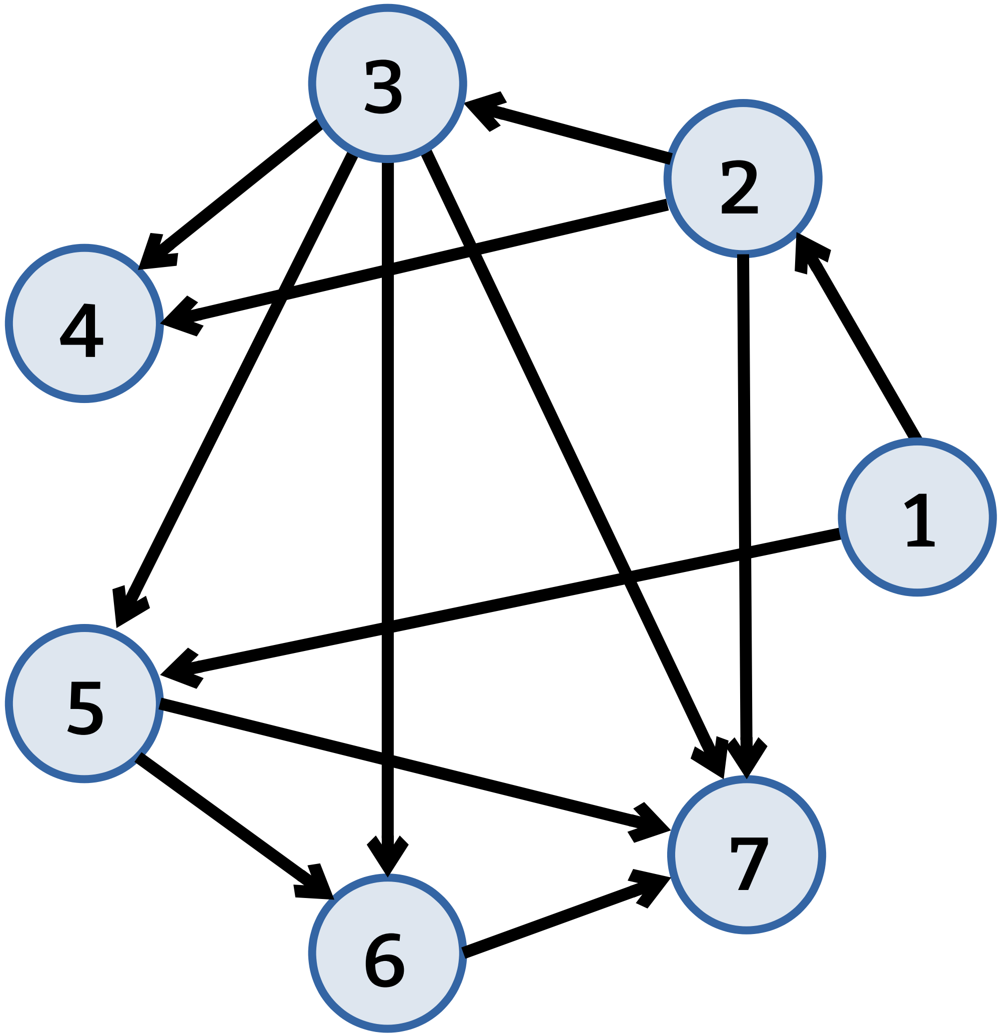
Directed graph



Directed acyclic graph (DAG)

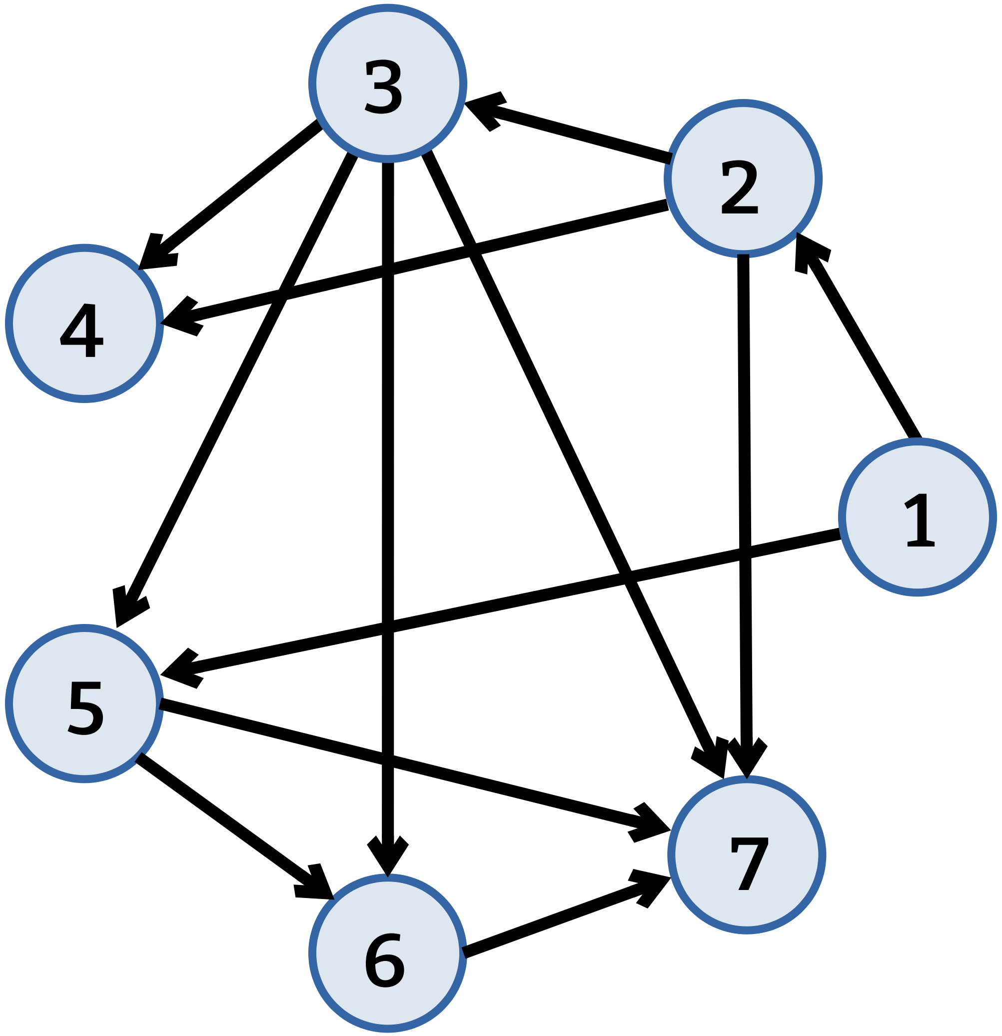


Adjacency Matrix



	1	2	3	4	5	6	7
1	0	1	0	0	0	0	0
2	0	0	1	0	0	0	1
3	0	0	0	1	1	1	1
4	0	0	0	0	0	0	0
5	0	0	0	0	0	1	1
6	0	0	0	0	0	0	1
7	0	0	0	0	0	0	0

Topological order



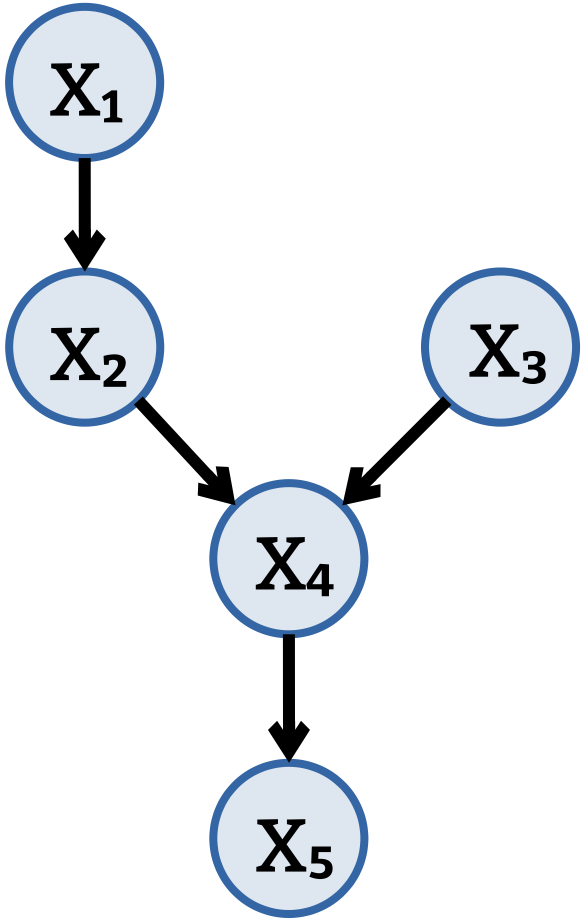
Data Generation Process

$X_1 = f_1(\varepsilon_1)$
$X_2 = f_2(X_1, \varepsilon_2)$
$X_3 = f_3(\varepsilon_3)$
$X_4 = f_4(X_2, X_3, \varepsilon_4)$
$X_5 = f_5(X_4, \varepsilon_5)$

Structural Causal Model

$$X_1 = f_1(\varepsilon_1)$$
$$X_2 = f_2(X_1, \varepsilon_2)$$
$$X_3 = f_3(\varepsilon_3)$$
$$X_4 = f_4(X_2, X_3, \varepsilon_4)$$
$$X_5 = f_5(X_4, \varepsilon_5)$$

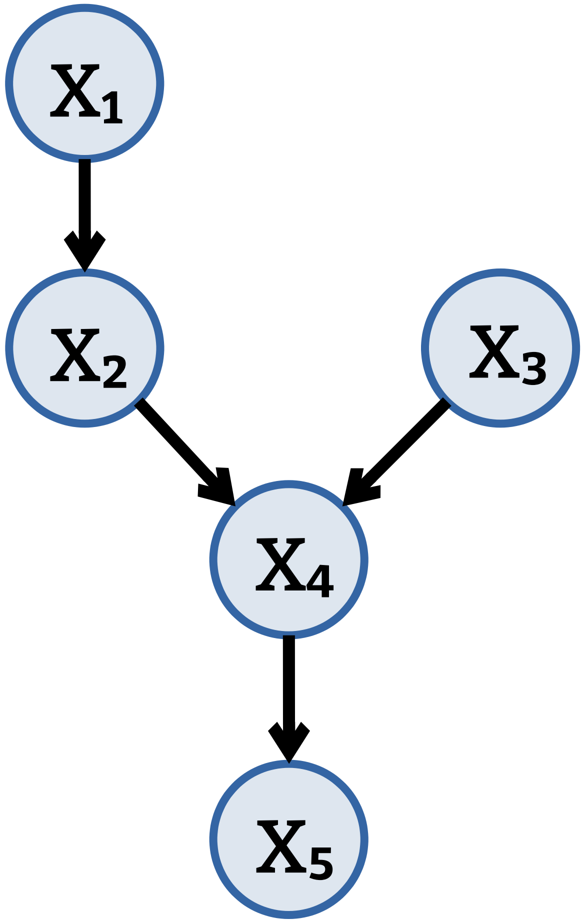
Causal Graph



Structural Causal Model

$$X_1 = f_1(\varepsilon_1)$$
$$X_2 = f_2(X_1, \varepsilon_2)$$
$$X_3 = f_3(\varepsilon_3)$$
$$X_4 = f_4(X_2, X_3, \varepsilon_4)$$
$$X_5 = f_5(X_4, \varepsilon_5)$$

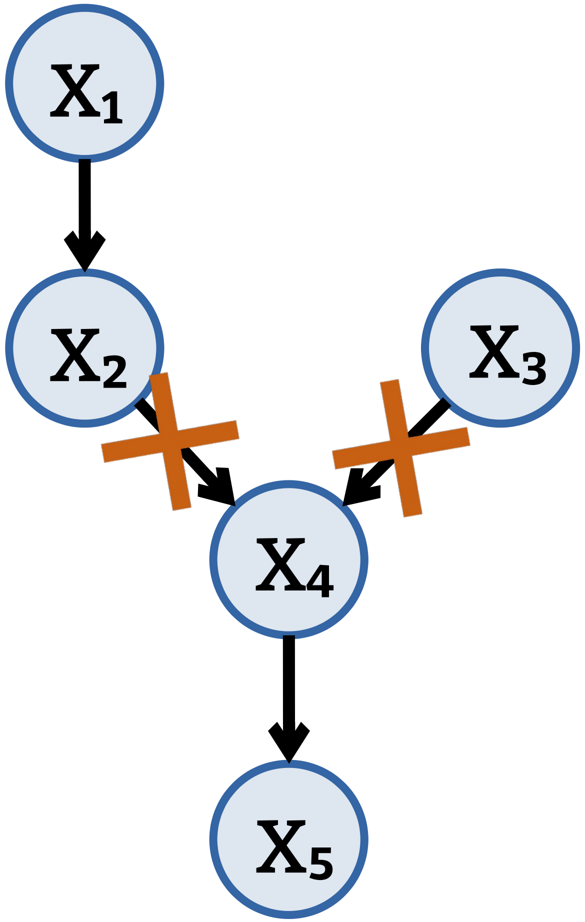
Causal Discovery



Causal Inference

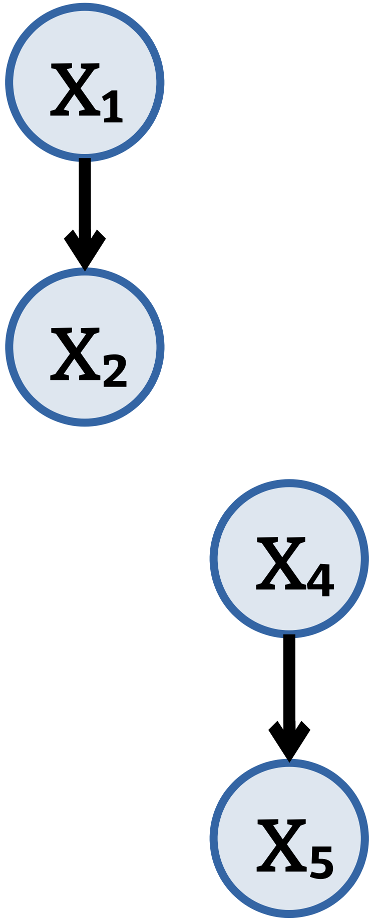
$$X_1 = f_1(\varepsilon_1)$$
$$X_2 = f_2(X_1, \varepsilon_2)$$
$$X_3 = f_3(\varepsilon_3)$$
$$X_4 = f_4(X_2, X_3, \varepsilon_4)$$
$$X_5 = f_5(X_4, \varepsilon_5)$$

Intervention



$X_1 = f_1(\varepsilon_1)$
$X_2 = f_2(X_1, \varepsilon_2)$
$X_3 = f_3(\varepsilon_3)$
$X_4 = x_4$
$X_5 = f_5(X_4, \varepsilon_5)$

Intervention



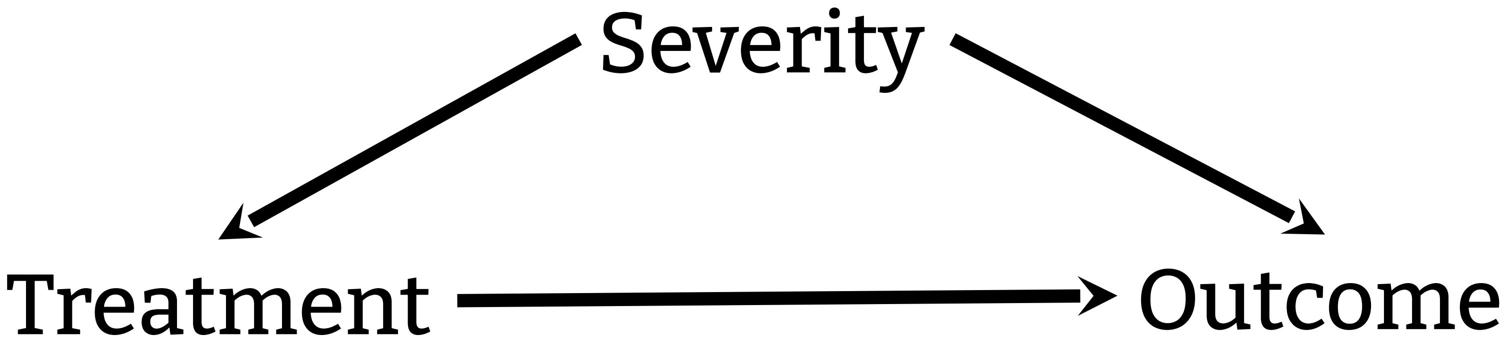
$$X_1 = f_1(\varepsilon_1)$$
$$X_2 = f_2(X_1, \varepsilon_2)$$
$$X_3 = f_3(\varepsilon_3)$$
$$X_4 = x_4$$
$$X_5 = f_5(X_4, \varepsilon_5)$$

Discovery vs Inference

- **Causal discovery**: finding the causal graph from data
- **Causal inference**: using the structural causal model to answer “what if” questions

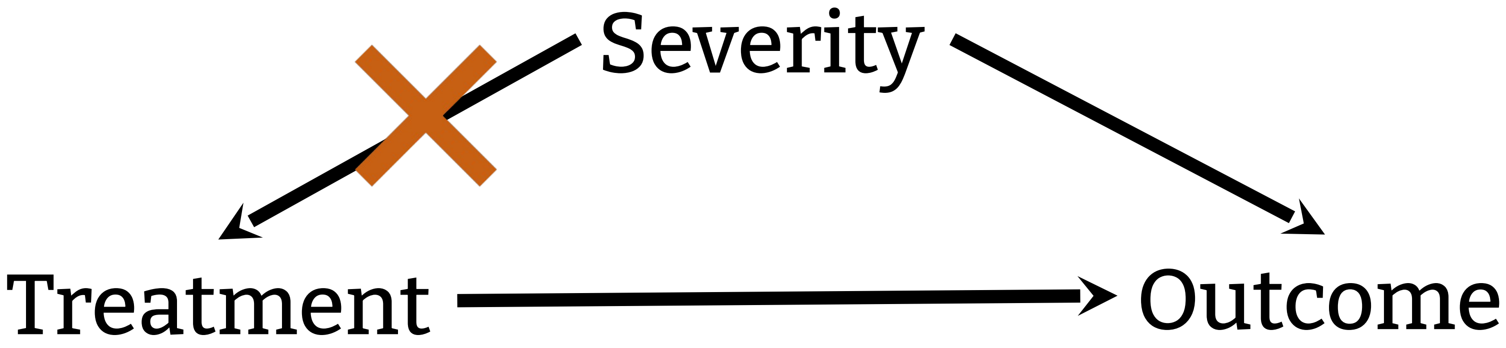
Causality ladder

- **Association**: E[Y|T=t]
- Intervention: E[Y|do(T=t)]
- Counterfactuals: E[Y|do(T=t),T=t’]



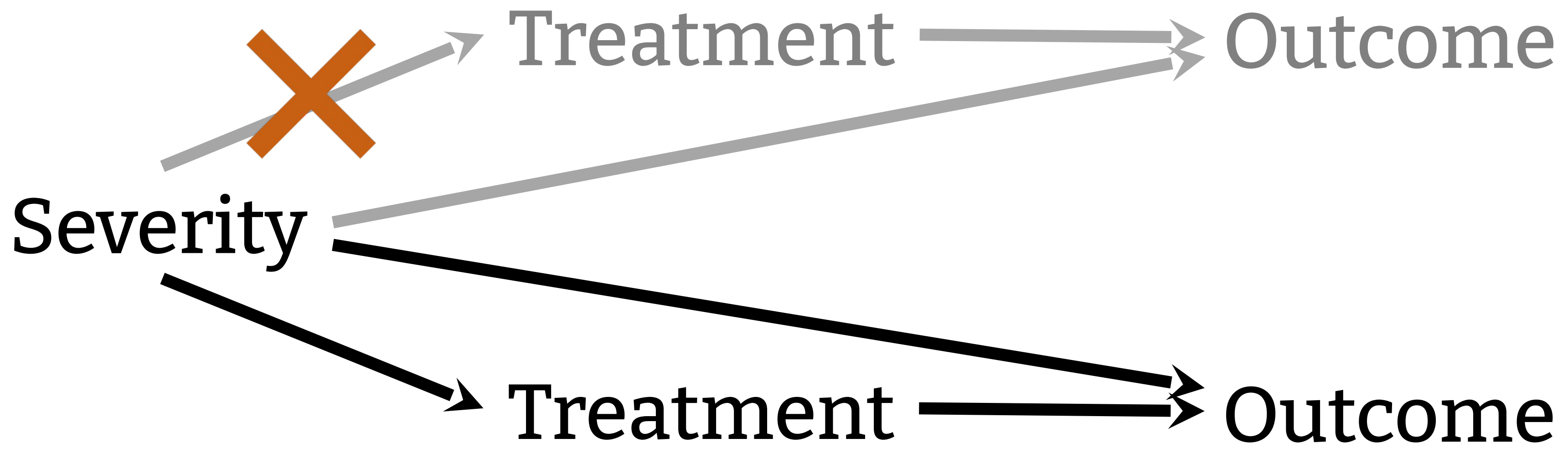
Causality ladder

- Association: $E[Y|T=t]$
- **Intervention**: $E[Y|do(T=t)]$
- Counterfactuals: $E[Y|do(T=t),T=t']$



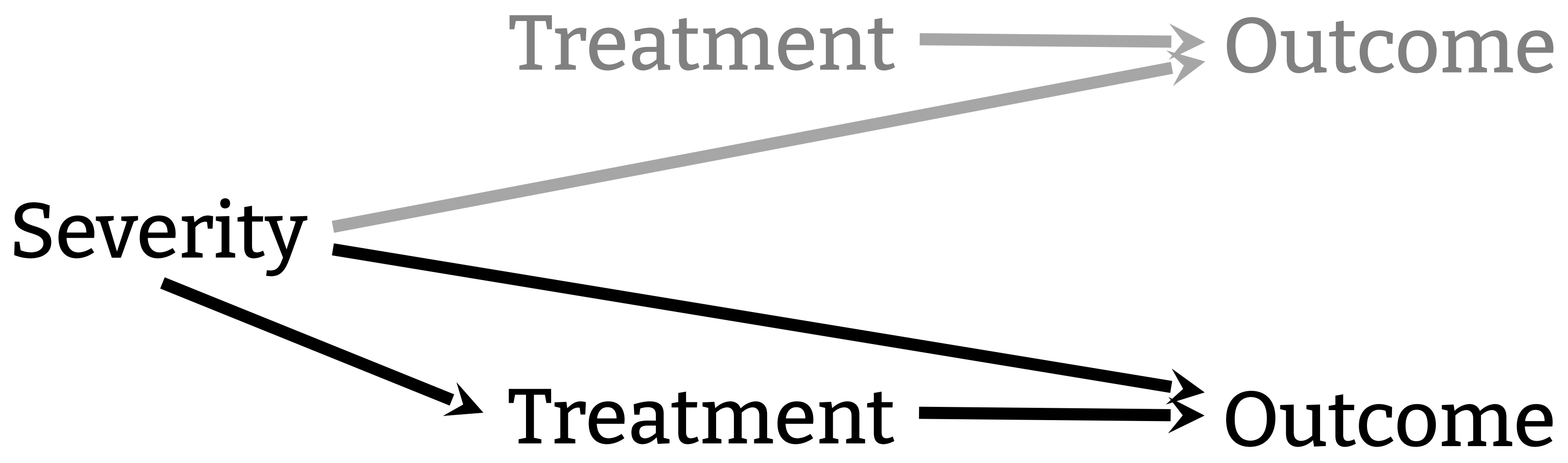
Causality ladder

- **Association**: $E[Y|T=t]$
- **Intervention**: $E[Y|do(T=t)]$
- **Counterfactuals**: $E[Y|do(T=t),T=t']$



Causality ladder

- Association: E[Y|T=t]
- Intervention: E[Y|do(T=t)]
- **Counterfactuals**: E[Y|do(T=t),T=t’]



Causality Ladder

- **Association**

  If the patient received an aggressive treatment, it means his condition was already severe: the expected outcome is bad.

- **Intervention**

  If we were to give all patients an aggressive treatment, the outcome would be good on average.

- **Counterfactural**

  This specific patient received the non-aggressive treatment; this means his condition was mild; if we had given him the aggressive treatment, the outcome would have been good.

Simpson's paradox

Treatment	Condition: Mild	Condition: Severe	Condition: Total
A	15% (210/1400)	30% (30/100)	**16%** (240/1500)
B	**10%** (5/50)	**20%** (100/500)	19% (105/550)

Introduction to Causal Inference (B. Neal, 2020)

Fundamental Problem

We often want to compute the average treatment effect,

$$ATE = E[Y|do(T=1)] - E[Y|do(T=0)],$$

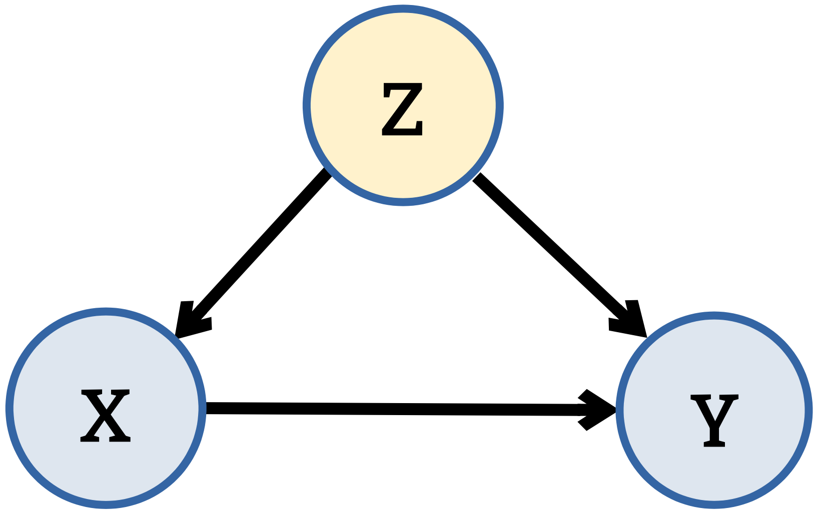
but, for each subject, we either have T=1 or T=0, so we do not know the other.

Do Calculus

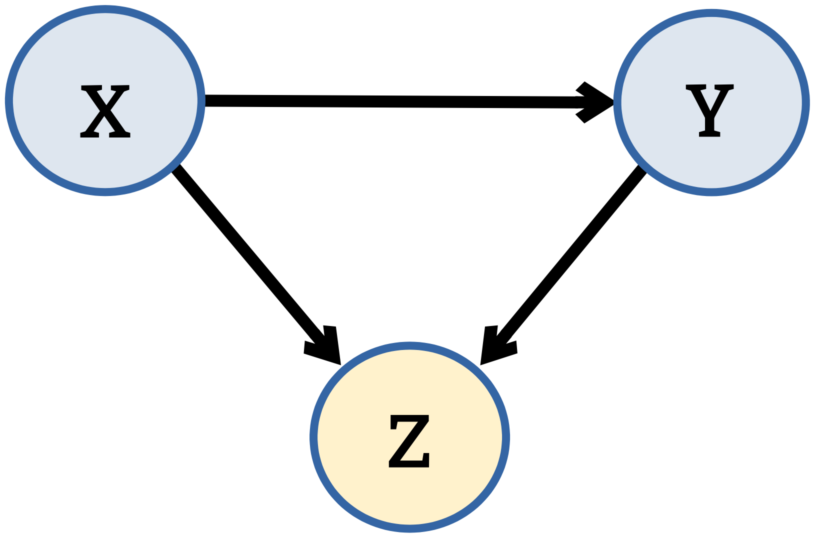
Do Calculus is a set of rules to compute, when possible, the effect interventions would have from observational data alone, given the causal graph.

Introduction to Causal Inference (B. Neal, 2020, theorem 6.2, page 55)

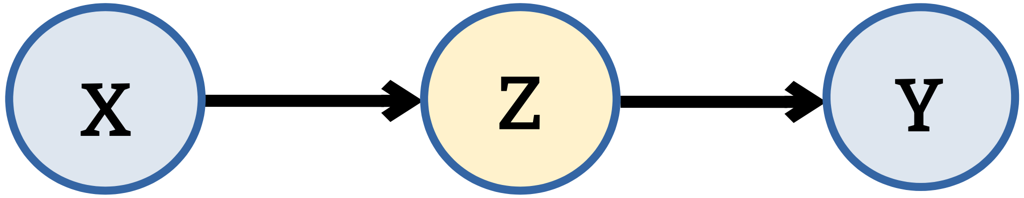
Confounder



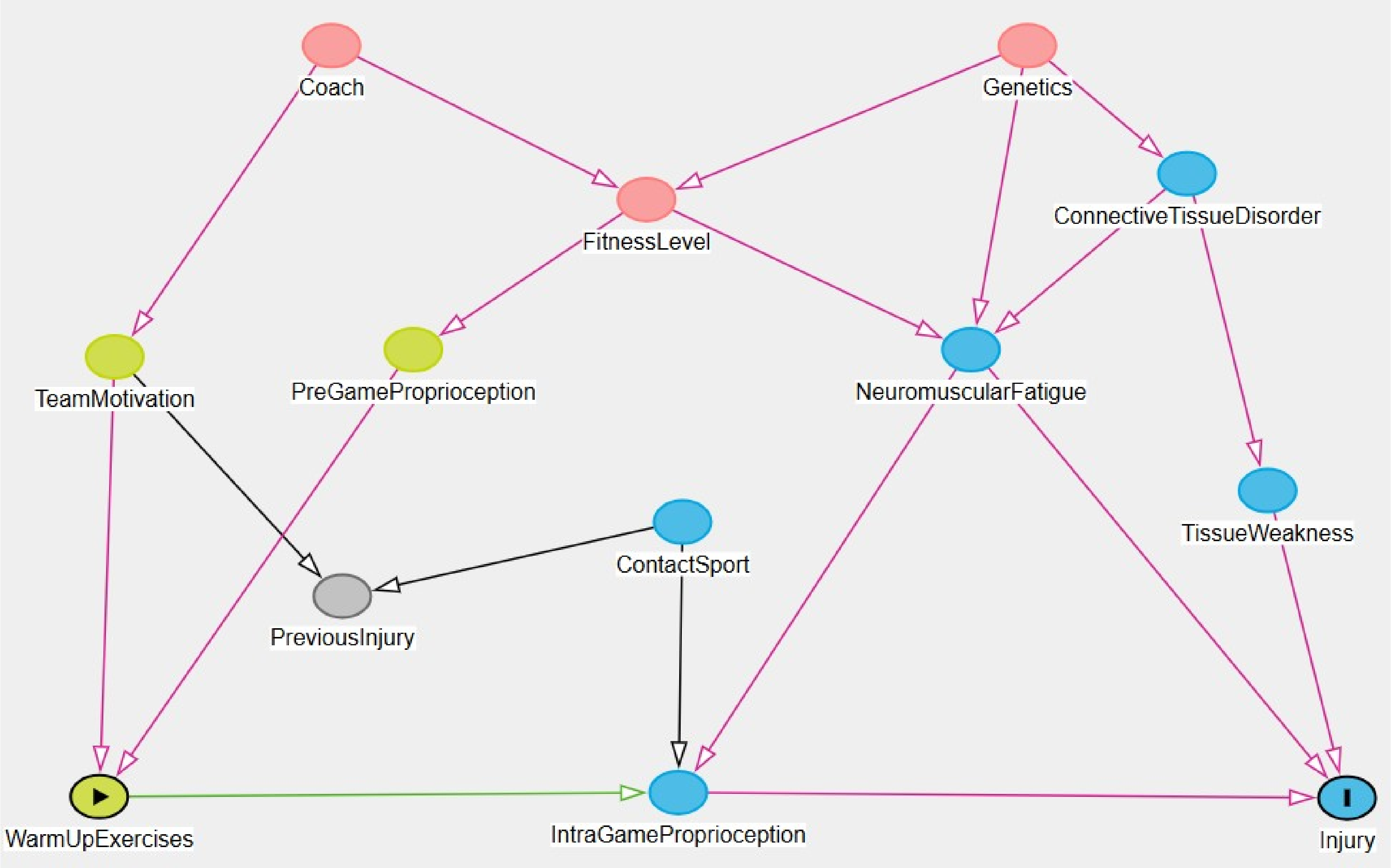
Collider



Mediator

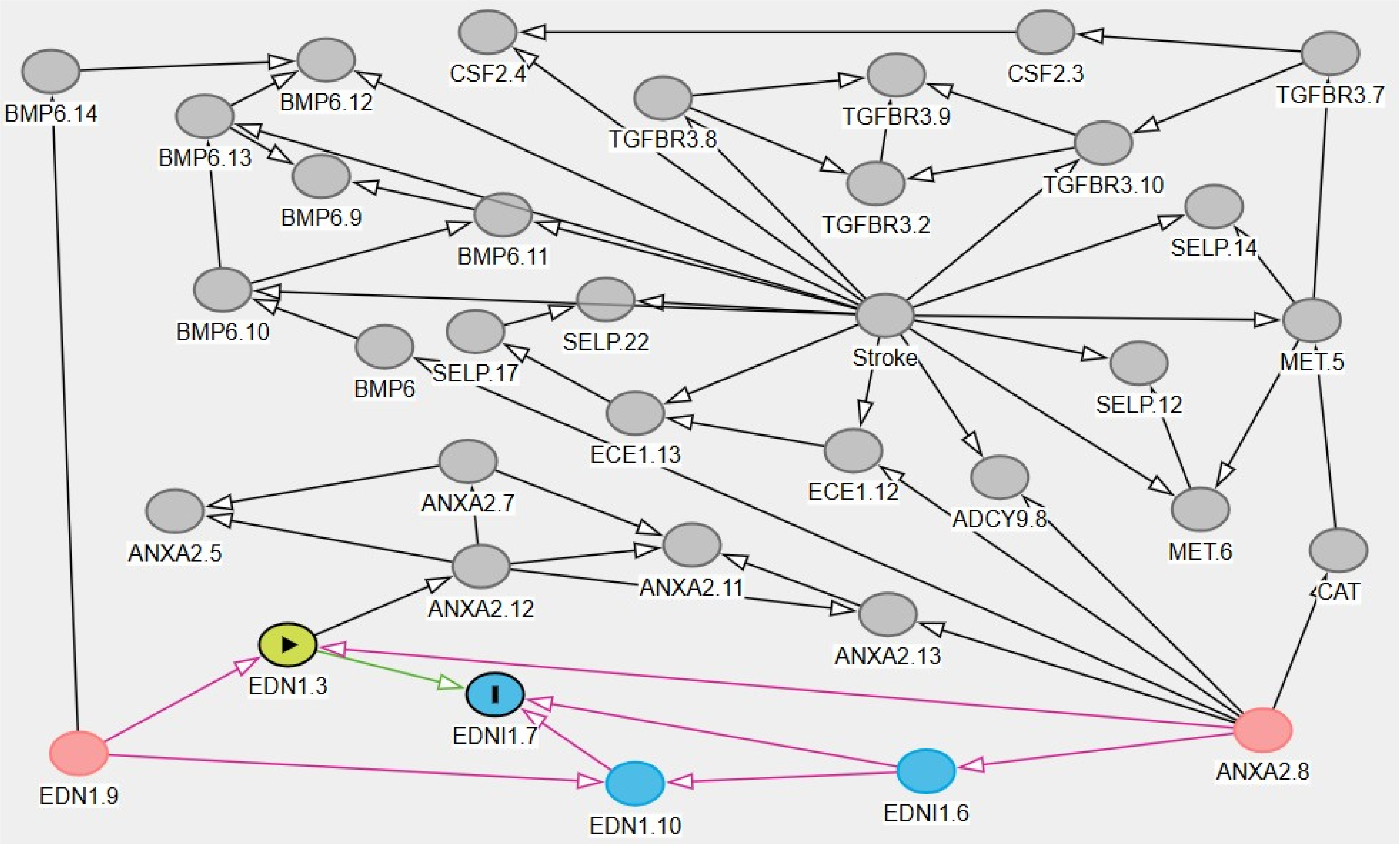


Real-world causal graphs



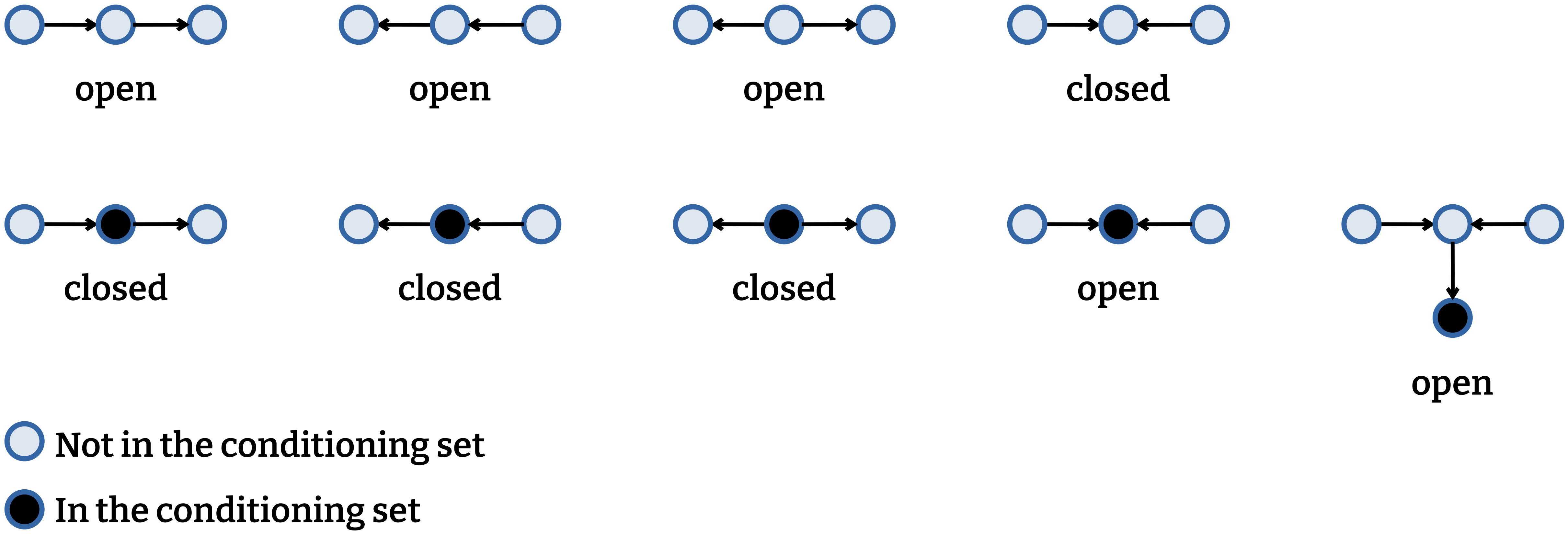
https://www.dagitty.net/dags.html

Real-world causal graphs



https://www.dagitty.net/dags.html

Open and closed paths



Causal inference

To assess the strength of the causal relation X→Y:

- List all the (undirected) paths from X to Y
- All the non-causal paths should be blocked; if not, condition on one or more nodes to block them.
- All the causal paths should be open; if not, adjust the conditioning set to unblock them.

Causal Discovery Algorithms

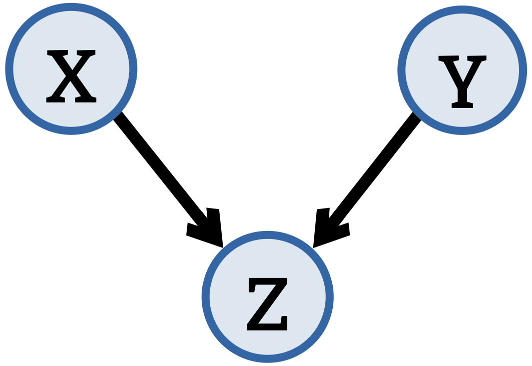
Causal Discovery Algorithms

- **PC**: Conditional independence tests
- **GES**: Scores
- **LiNGAM**: Independent component analysis
- **NOTEARS**: Optimization
- Deep Learning
- LLMs

PC Algorithm

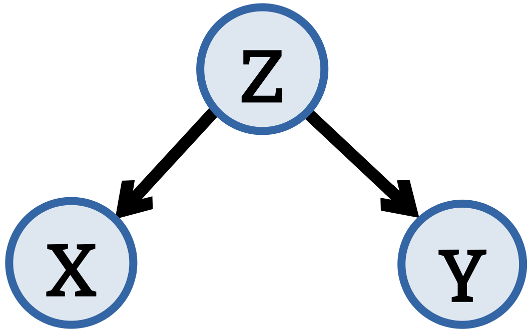
Conditional Independence

$X \not\perp Y$



$X \perp Y$

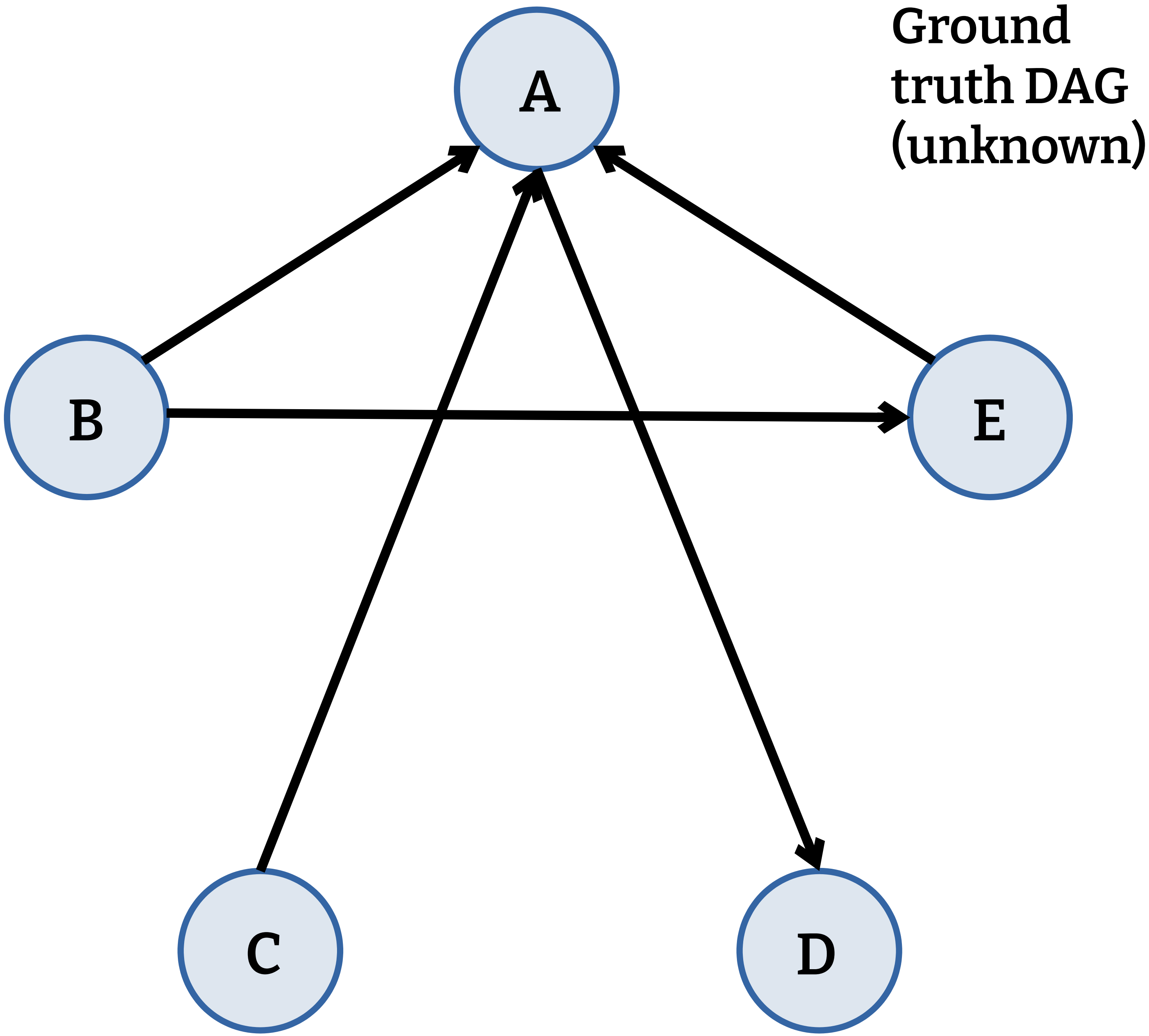
$X \not\perp Y | Z$

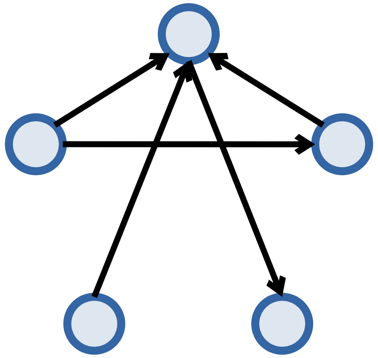


$X \not\perp Y$

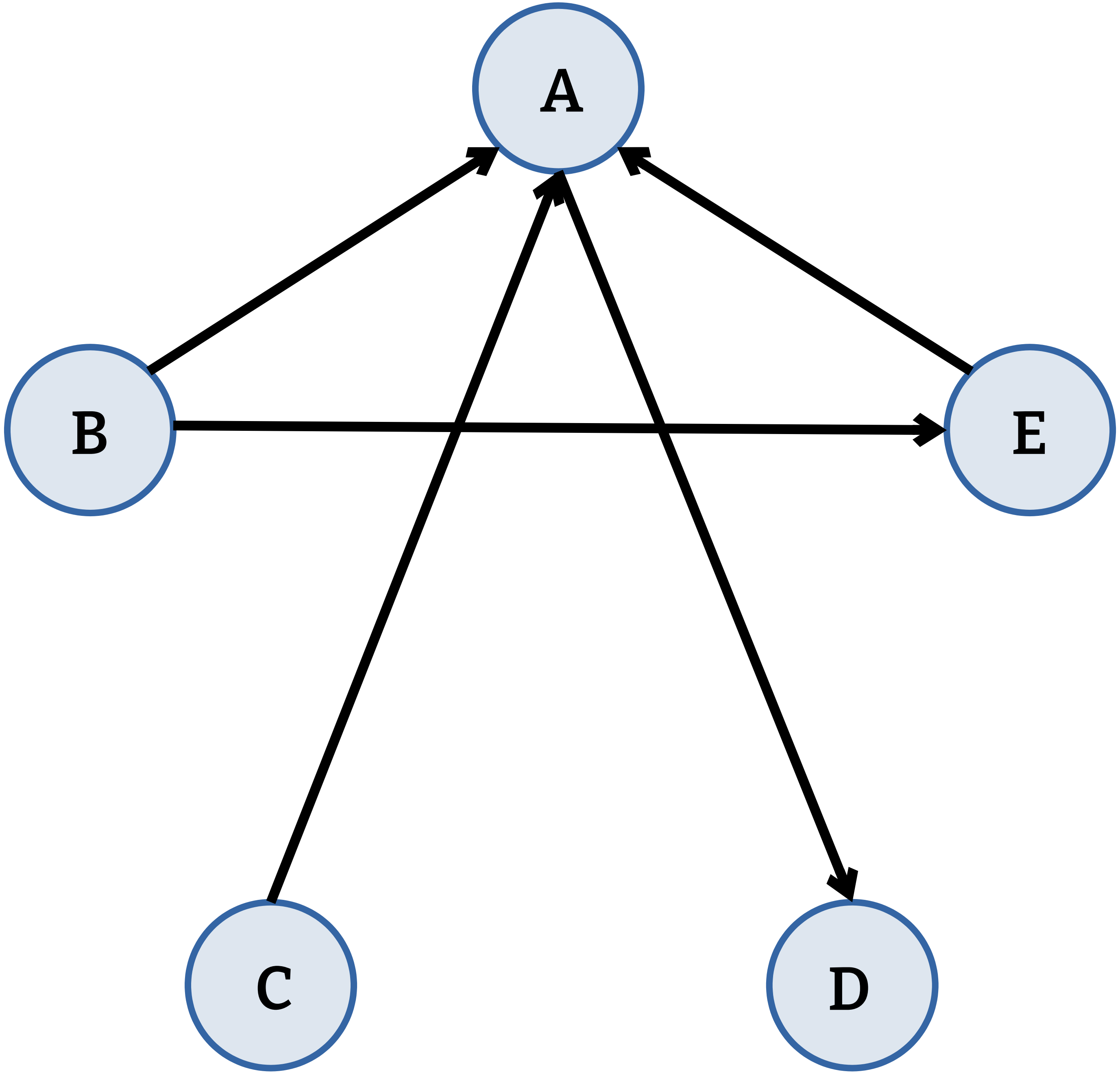
$X \perp Y | Z$

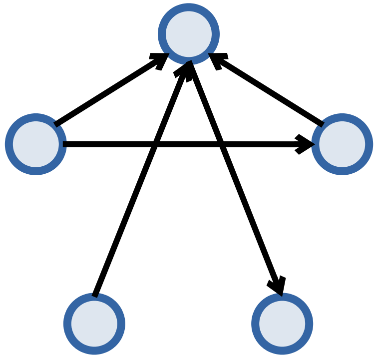
- Start with a complete (undirected) graph
- Remove the edge X—Y if $X \perp Y | Z$ for some (possibly empty) set of nodes Z
- For all X—Z—Y, if $X \perp Y$ and $X \not\perp Y | Z$, we have a collider $X \rightarrow Z \leftarrow Y$
- Propagate the orientation, assuming we have found all the colliders



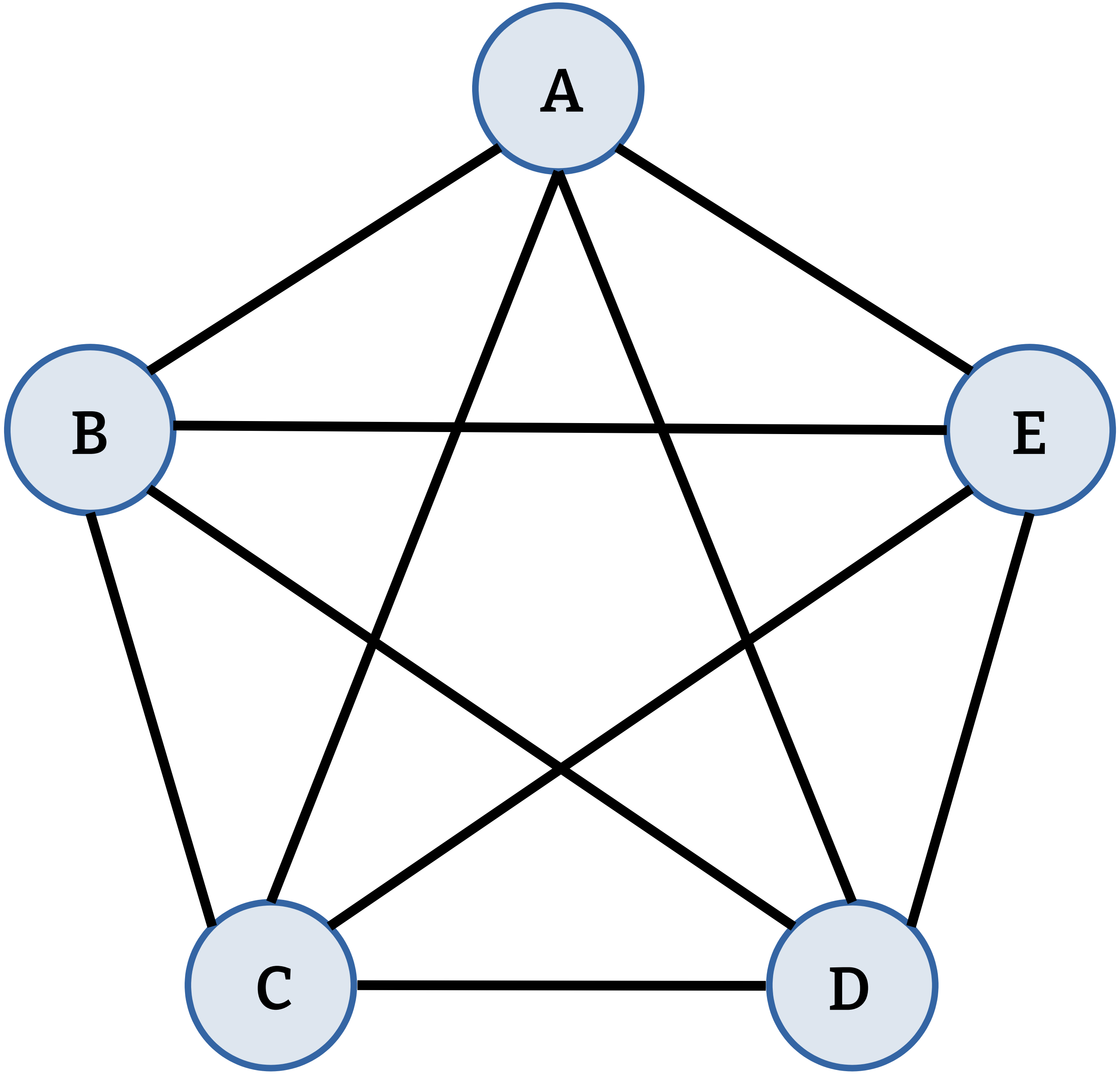


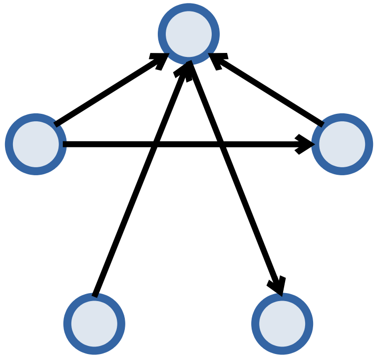
- Start with a complete (undirected) graph
- Remove the edge X—Y if $X \perp Y | Z$ for some (possibly empty) set of nodes Z
- For all X—Z—Y, if $X \perp Y$ and $X \not\perp Y | Z$, we have a collider $X \rightarrow Z \leftarrow Y$
- Propagate the orientation, assuming we have found all the colliders





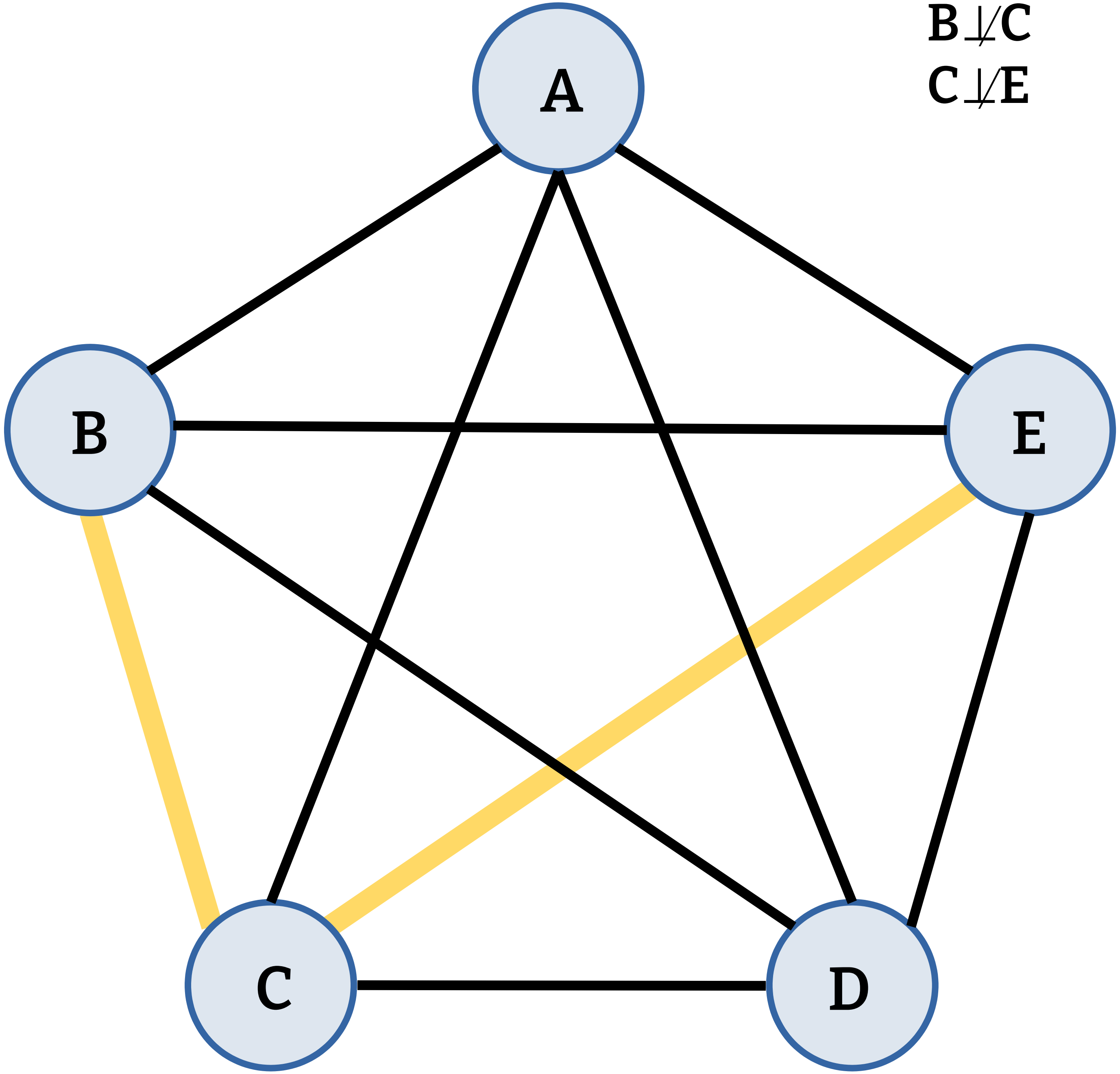
- **Start with a complete (undirected) graph**
- Remove the edge X—Y if $X \perp Y | Z$ for some (possibly empty) set of nodes Z
- For all X—Z—Y, if $X \perp Y$ and $X \not\perp Y | Z$, we have a collider X→Z←Y
- Propagate the orientation, assuming we have found all the colliders

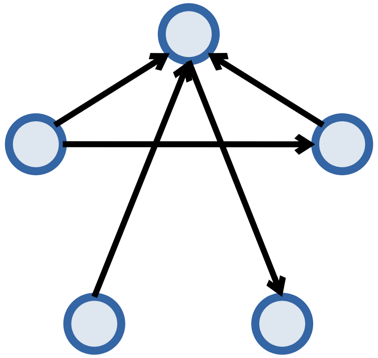




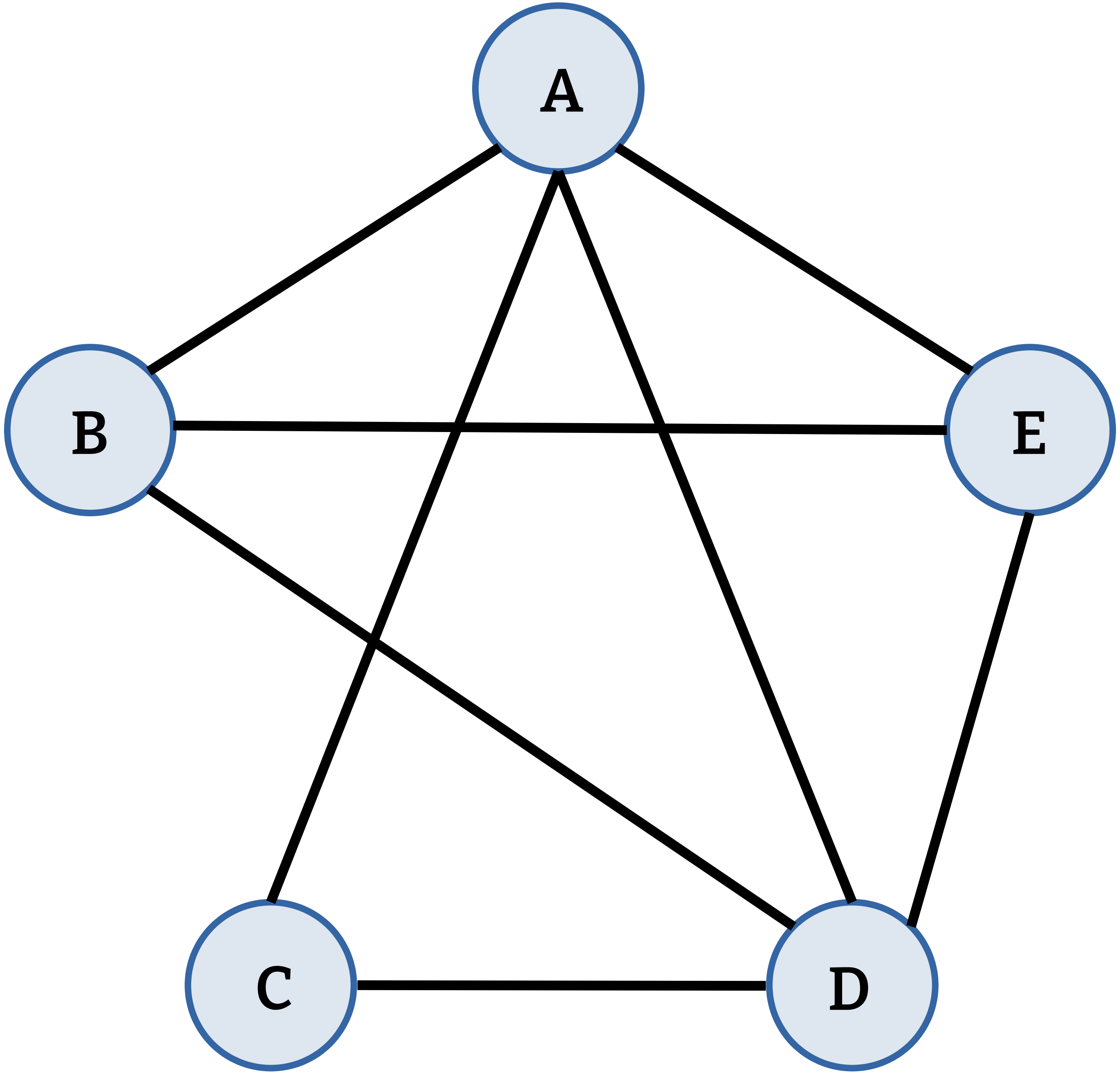
- Start with a complete (undirected) graph
- Remove the edge X—Y if $X \perp Y | Z$ for some (possibly empty) set of nodes Z
- For all X—Z—Y, if $X \perp Y$ and $X \not\perp Y |$ Z, we have a collider X→Z←Y
- Propagate the orientation, assuming we have found all the colliders

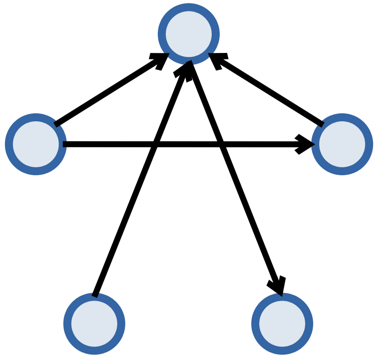
$B \not\perp C$
$C \not\perp E$



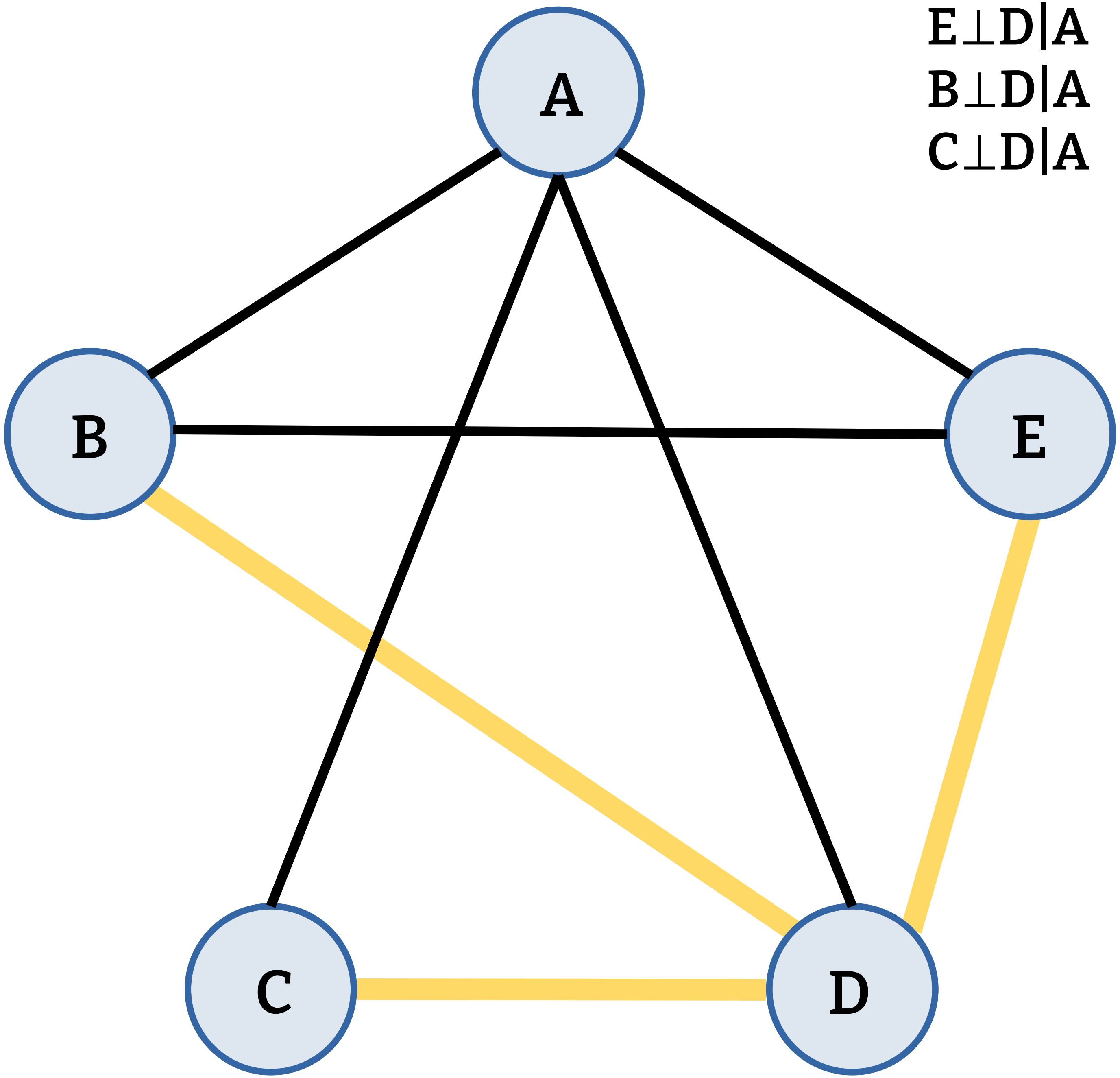


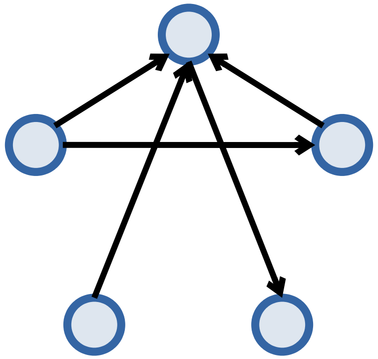
- Start with a complete (undirected) graph
- Remove the edge X—Y if $X \perp Y | Z$ for some (possibly empty) set of nodes Z
- For all X—Z—Y, if $X \perp Y$ and $X \not\perp Y | Z$, we have a collider X→Z←Y
- Propagate the orientation, assuming we have found all the colliders



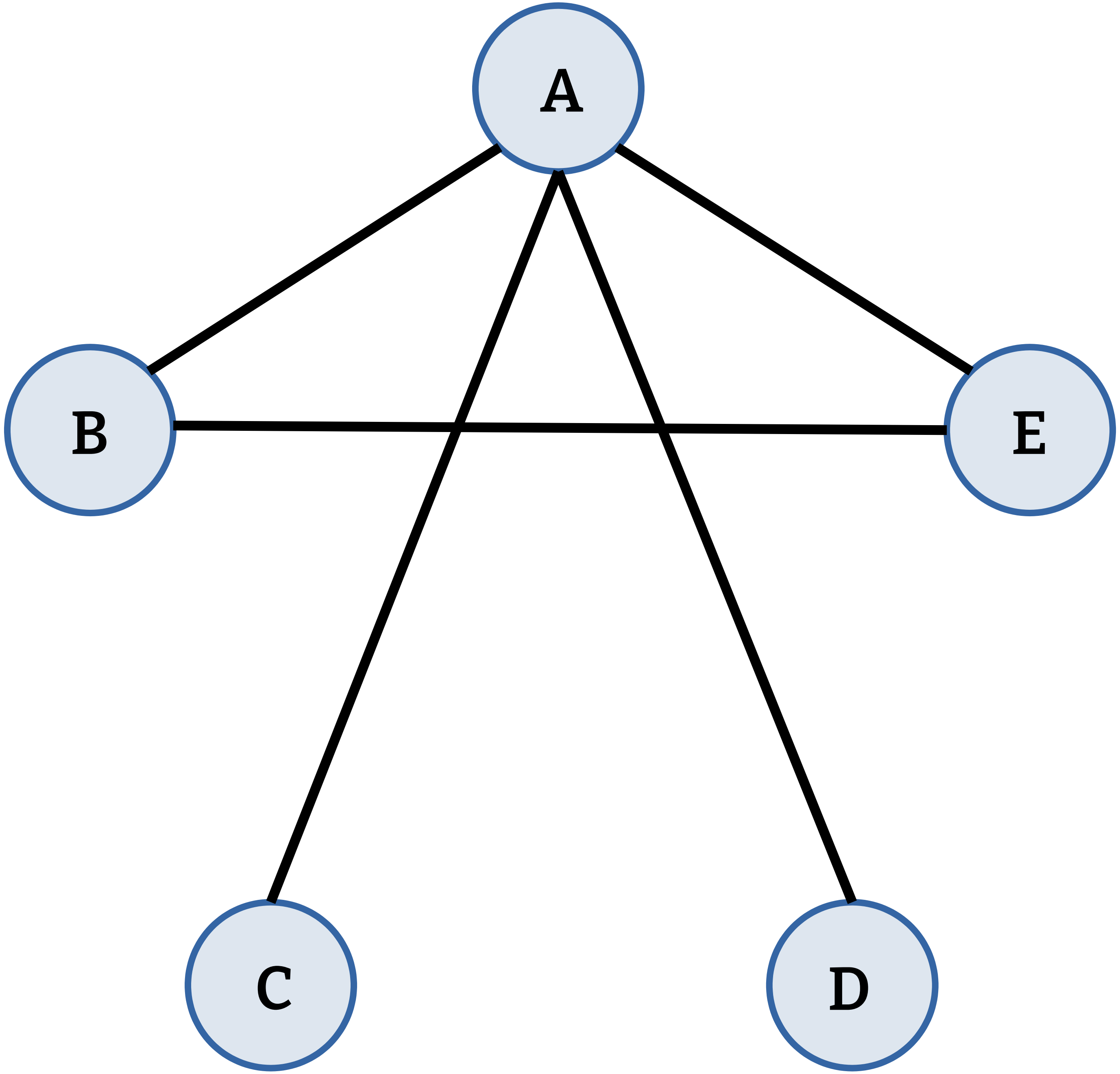


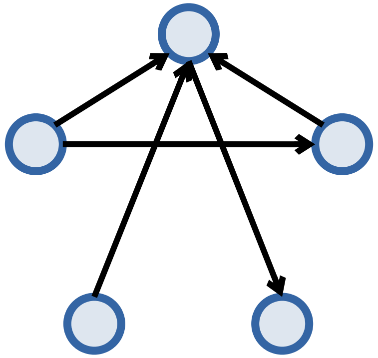
- Start with a complete (undirected) graph
- Remove the edge X—Y if X⊥Y|Z for some (possibly empty) set of nodes Z
- For all X—Z—Y, if X⊥Y and X⊥̸Y|Z, we have a collider X→Z←Y
- Propagate the orientation, assuming we have found all the colliders



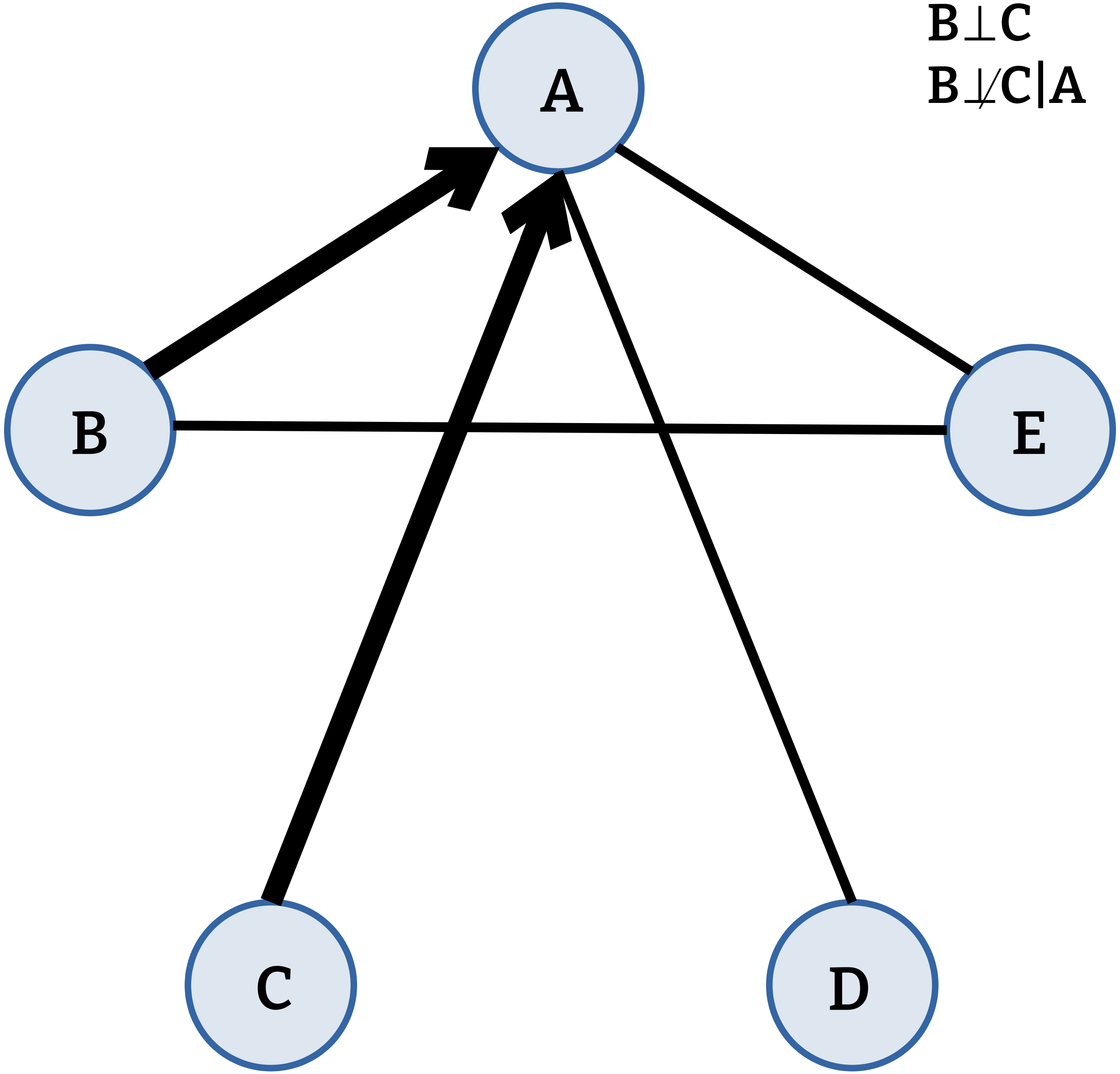


- Start with a complete (undirected) graph
- Remove the edge X—Y if $X \perp Y | Z$ for some (possibly empty) set of nodes Z
- For all X—Z—Y, if $X \perp Y$ and $X \not\perp Y | Z$, we have a collider $X \rightarrow Z \leftarrow Y$
- Propagate the orientation, assuming we have found all the colliders

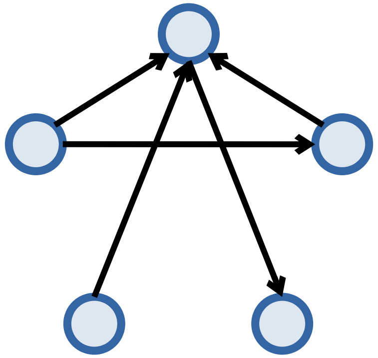




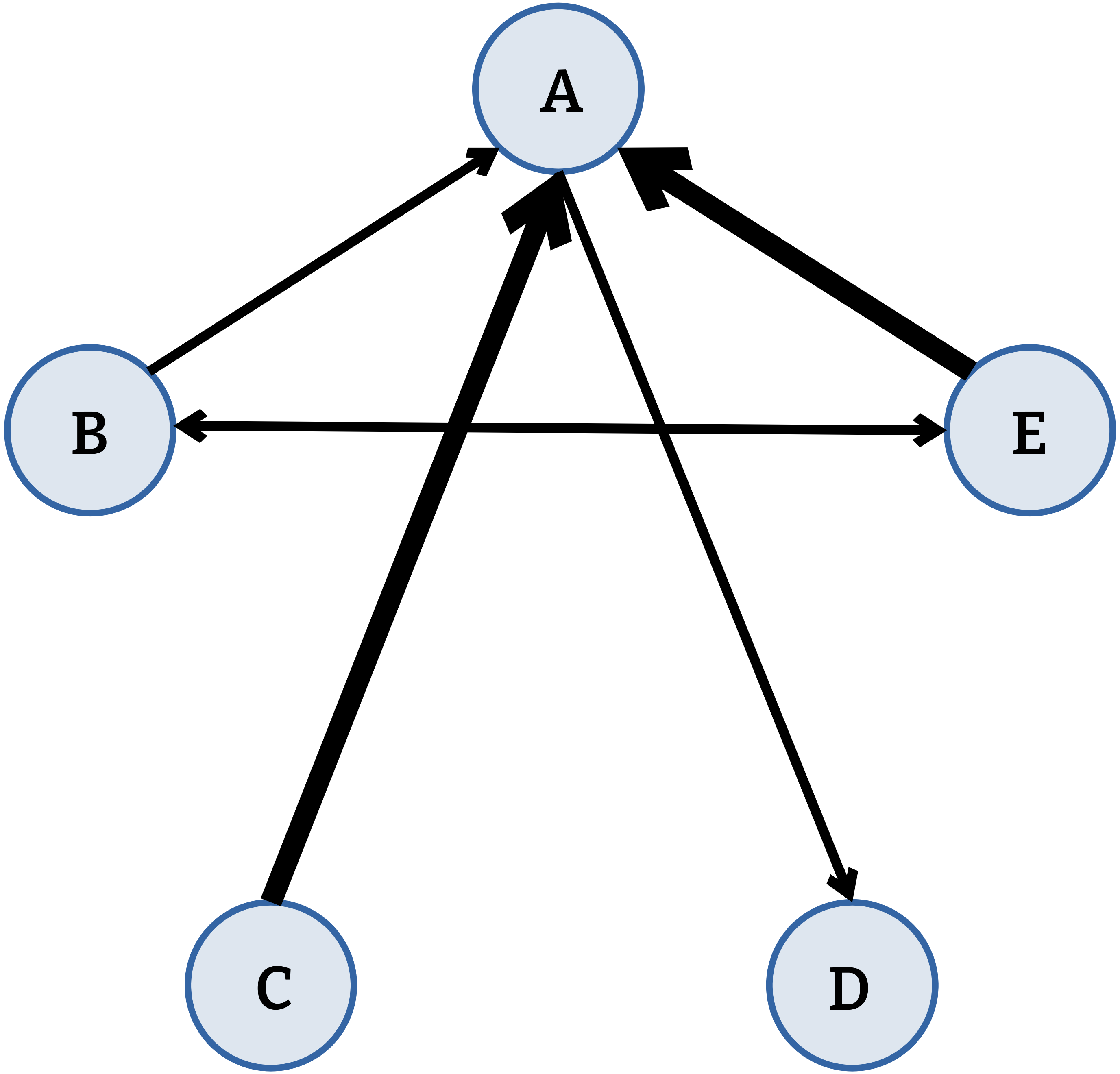
- Start with a complete (undirected) graph
- Remove the edge X—Y if X⊥Y|Z for some (possibly empty) set of nodes Z
- For all X—Z—Y, if X⊥Y and X⊥̸Y|Z, we have a collider X→Z←Y
- Propagate the orientation, assuming we have found all the colliders

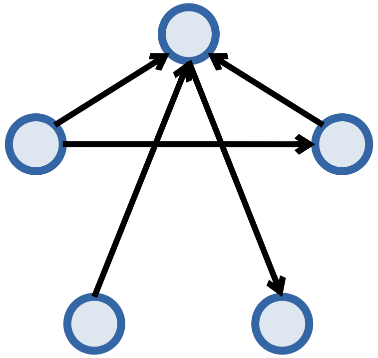


B⊥C
B⊥̸C|A

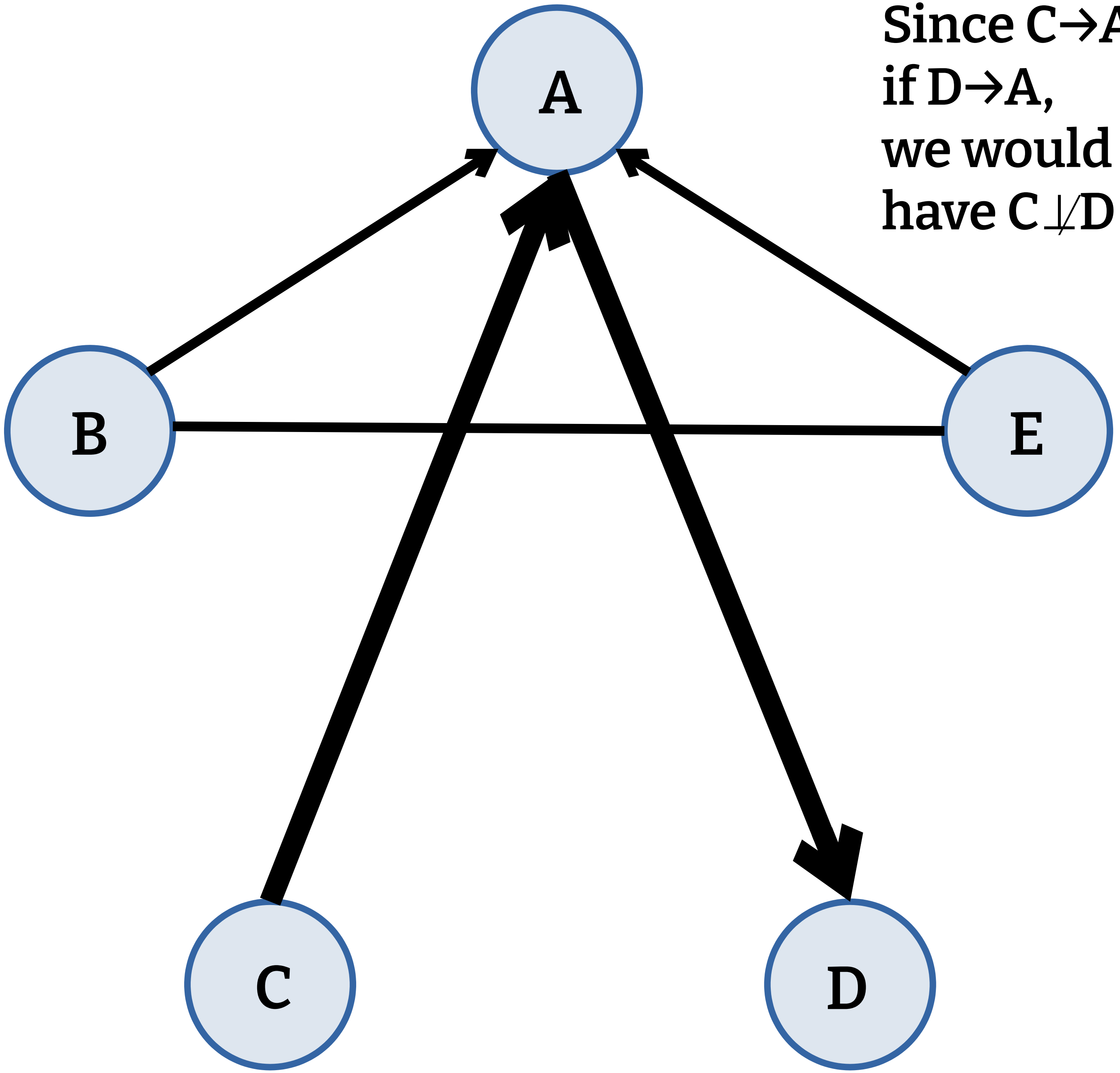


- Start with a complete (undirected) graph
- Remove the edge X—Y if X⊥Y|Z for some (possibly empty) set of nodes Z
- For all X—Z—Y, if X⊥Y and X⊥̸Y|Z, we have a collider X→Z←Y
- Propagate the orientation, assuming we have found all the colliders

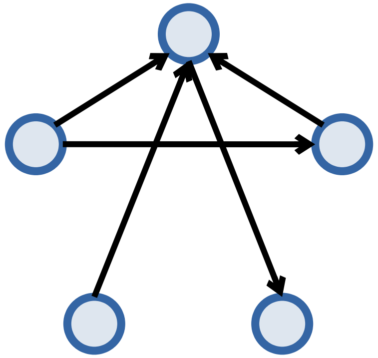




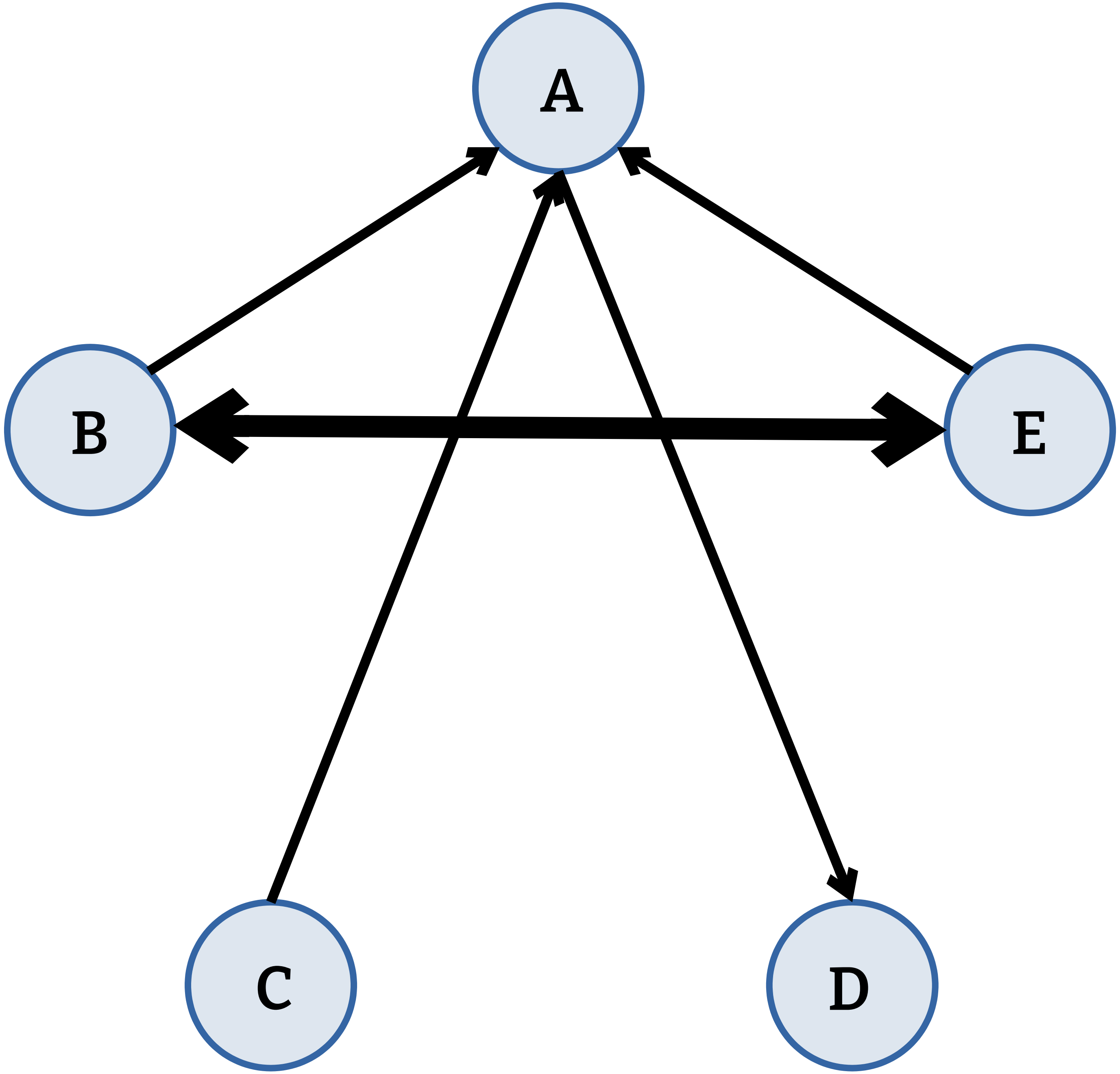
- Start with a complete (undirected) graph
- Remove the edge X—Y if $X \perp Y | Z$ for some (possibly empty) set of nodes Z
- For all X—Z—Y, if $X \perp Y$ and $X \not\perp Y | Z$, we have a collider X→Z←Y
- **Propagate the orientation, assuming we have found all the colliders**



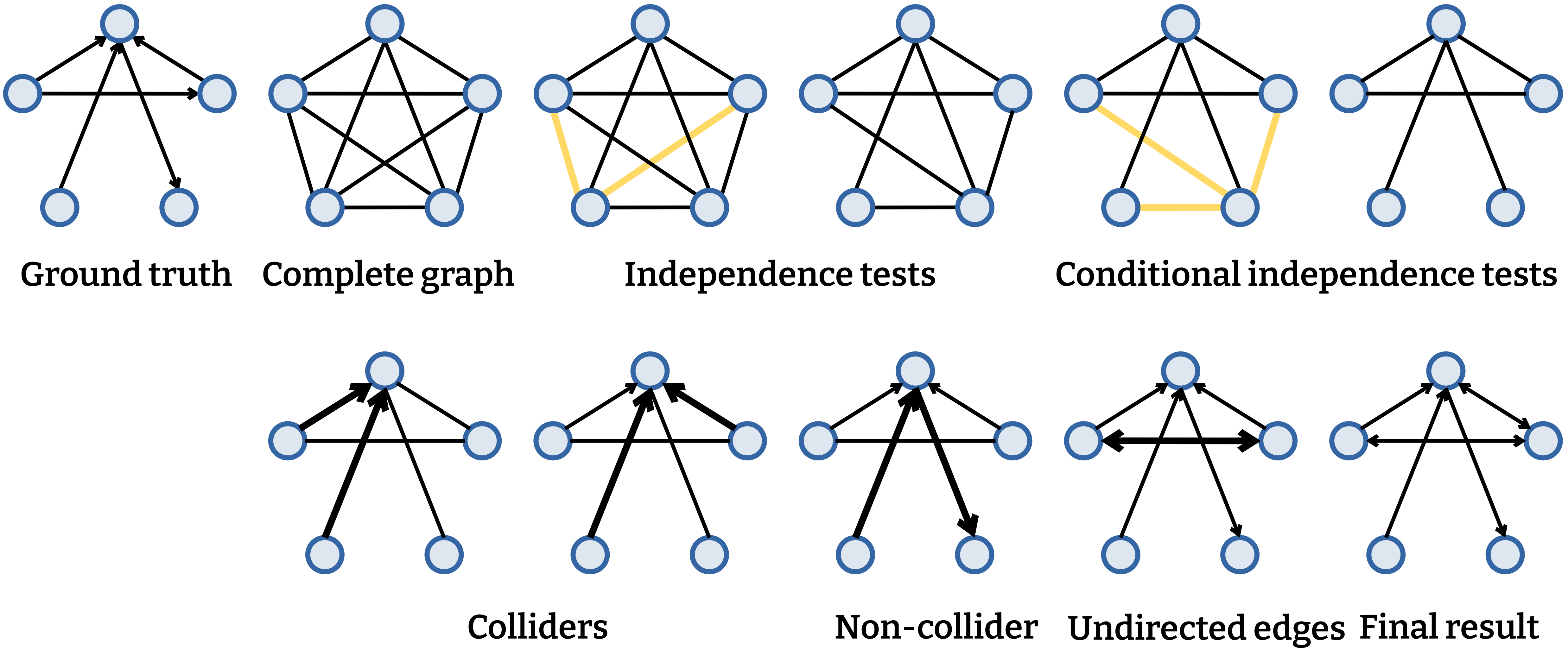
Since C→A,
if D→A,
we would
have $C \not\perp D | A$



- Start with a complete (undirected) graph
- Remove the edge X—Y if $X \perp Y | Z$ for some (possibly empty) set of nodes Z
- For all X—Z—Y, if $X \perp Y$ and $X \not\perp Y | Z$, we have a collider X→Z←Y
- Propagate the orientation, assuming we have found all the colliders

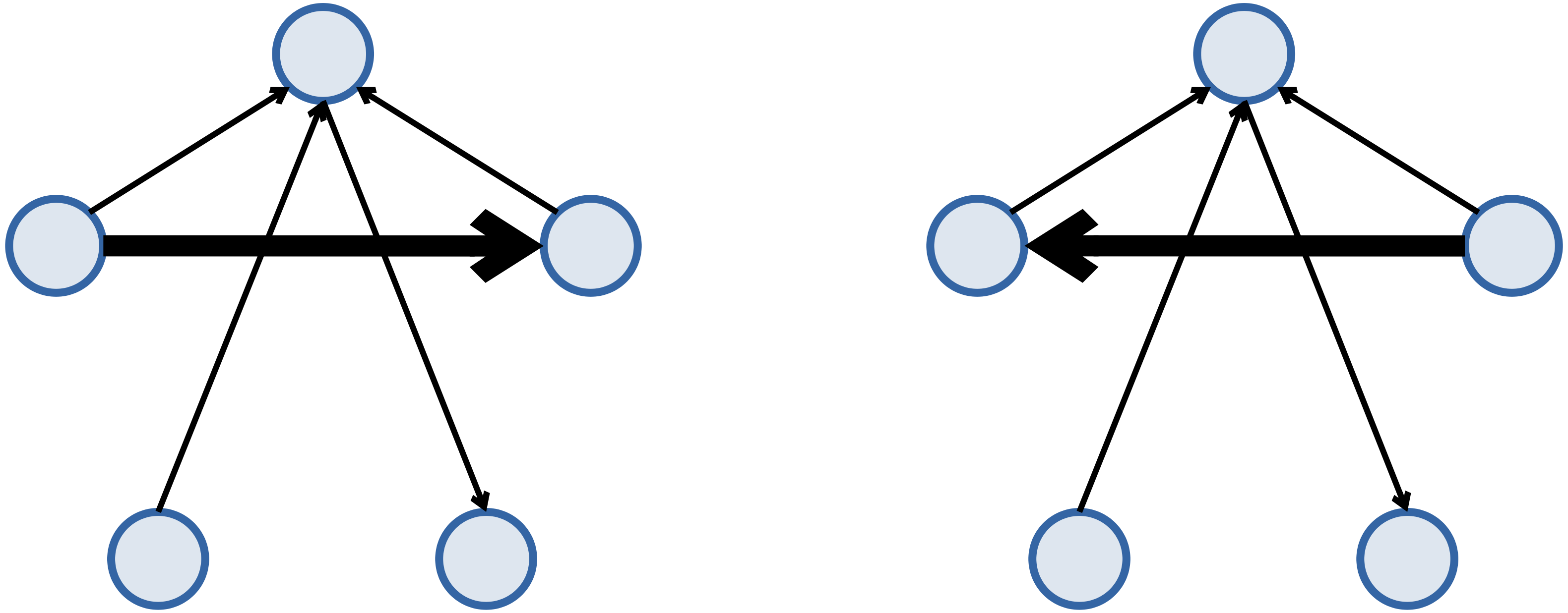


PC algorithm



Markov equivalence class

Conditional independence relations cannot separate all DAGs: they can only recover the Markov equivalence class (MEC) of the causal model.



Independence tests

- Partial correlation (Fisher Z)
- Kernel-based tests
- χ^2 test (for discrete data)

GES

GES (Greedy Equivalent Search)

- Start with an empty graph
- Greedily add the edges whose addition increases the score the most
- Greedily remove the edges whose removal increase the score the most
- Score: BIC, when forecasting a variable from its parents

LiNGAM

LinGAM

The structural causal model (SCM)

$$X_1 = \mu_1 + \varepsilon_1$$

$$X_2 = \mu_2 + a_{21} X_1 + \varepsilon_2$$

$$\vdots$$

$$X_k = \mu_k + a_{k1} + \cdots + a_{kk-1} X_{k-1} + \varepsilon_k$$

can be written $X = \mu + AX + \varepsilon$, or $\varepsilon = (I-A)X - \mu$, with $\forall i \neq j\ \varepsilon_i \perp \varepsilon_j$

If the ε_i's are non-Gaussian, ICA (independent component analysis) can recover the linear transformation (I-A)X by looking for the directions in which the data is the least Gaussian.

NOTEARS

NOTEARS

Find a matrix W such that WX≈X and the corresponding graph is acyclic.

The acyclicity condition can be written

trace exp |W| = n

(where |·| is the elementwise absolute value)

$$\exp A = I + A + \frac{1}{2!}A^2 + \frac{1}{3!}A^3 + \cdots$$

$$\operatorname{diag} A^k : \text{ number of cycles of length } k$$

NOTEARS

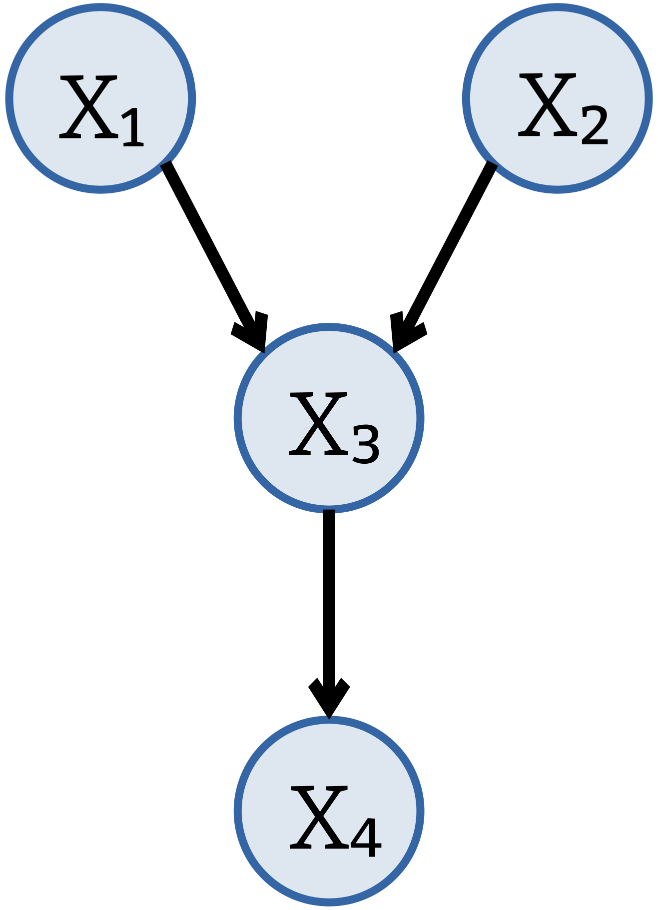
Find a matrix W such that WX≈X and the corresponding graph is acyclic.

$$
\begin{array}{ll}
\text{Find} & A \\
\text{To minimize} & \underset{i}{\text{Mean}} \left\| X_i - A X_i \right\|_F^2 + \lambda \left\| A \right\|_1 \\
\text{Such that} & \text{Trace}\, e^{|A|} = d
\end{array}
$$

Variants of those algorithms

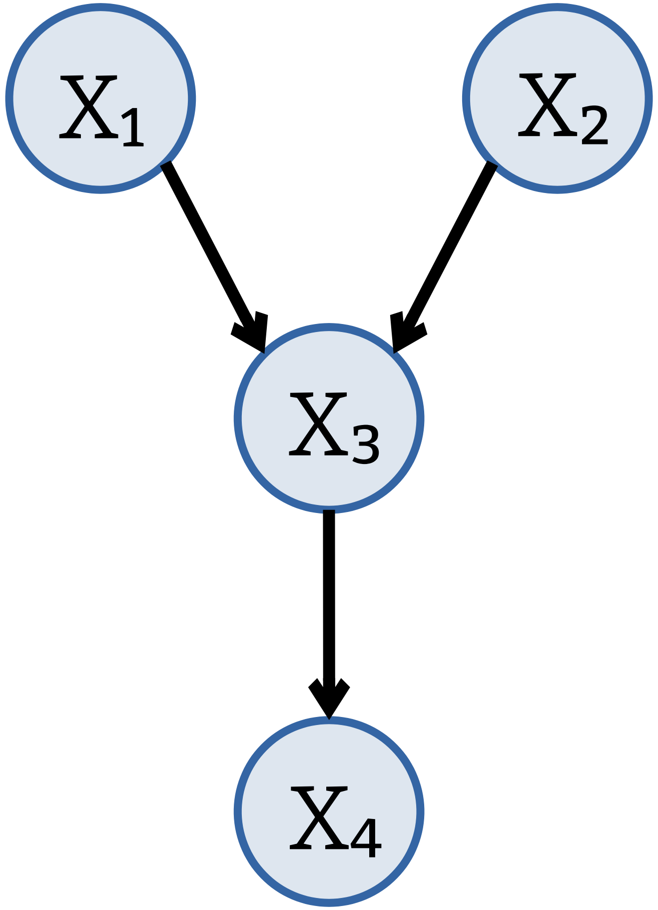
- Unobserved confounders
- Interventions
- Cycles
- Time series

Unobserved Confounders

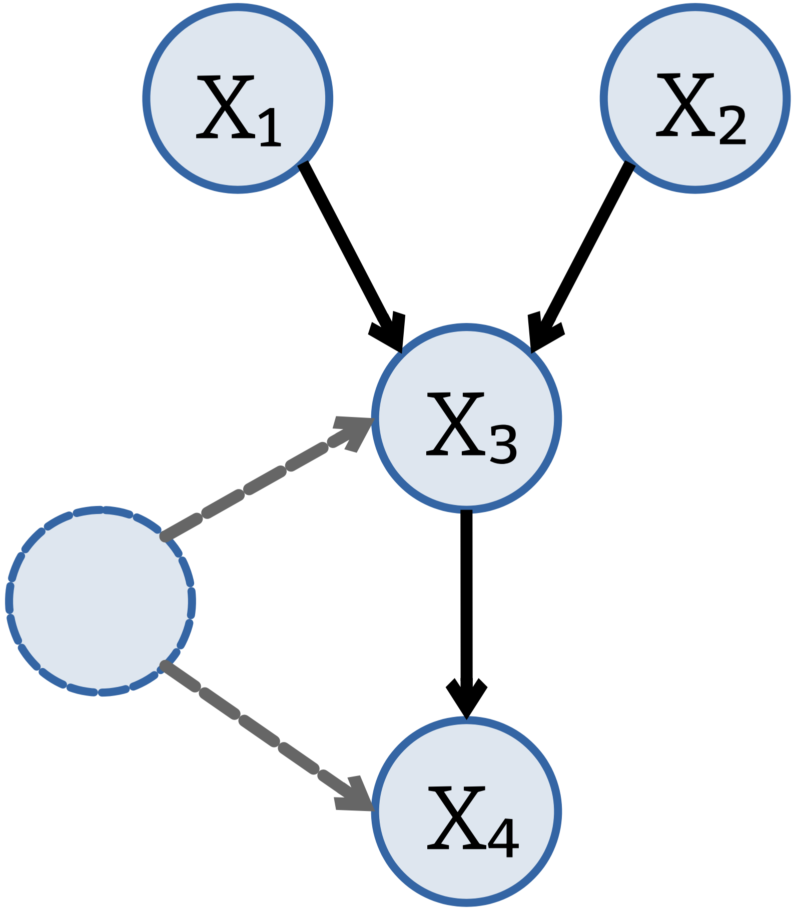


$$X_1 \perp X_4 \mid X_3$$

Unobserved Confounders

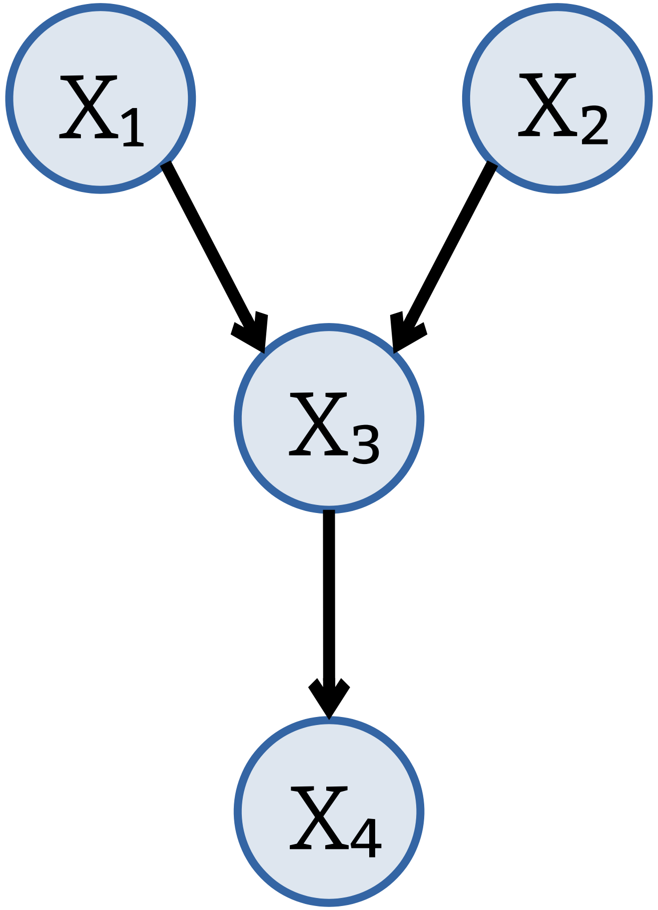


$$X_1 \perp X_4 \mid X_3$$



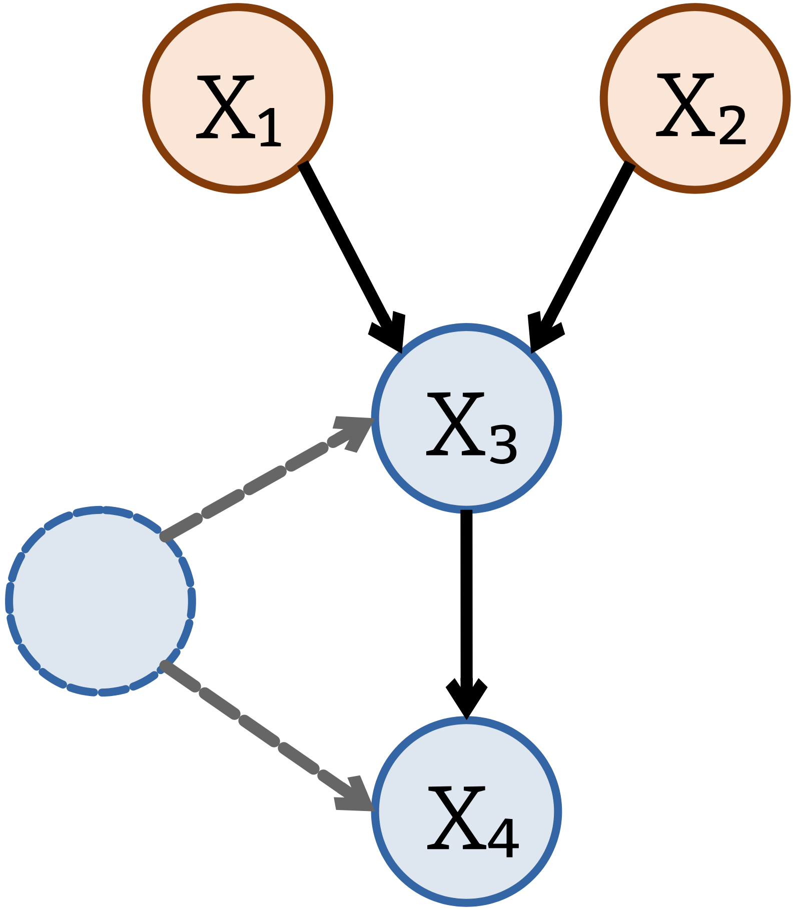
$$X_1 \not\perp X_4 \mid X_3$$

Unobserved Confounders



$X_1 \perp X_4 \mid X_3$

Instrumental variables



$X_1 \not\perp X_4 \mid X_3$

Deep Learning

Deep Learning

- Replace linear transformations with neural networks
- Better search algorithms for score-based methods (DP, A*)
- First look for a topological order
- Reinforcement learning for search
- Reinforcement learning to progressively build the solution

Beyond NOTEARS

- NOTEARS is linear:

$$
\begin{array}{ll}
\text{Find} & A \\
\text{To minimize} & \text{Mean}_i \left\| X_i - A X_i \right\|_F^2 + \lambda \left\| A \right\|_1 \\
\text{Such that} & \text{Trace}\, e^{A \odot A} = d
\end{array}
$$

A graph autoencoder approach to causal structure learning (I. Ng et al., 2019)

Beyond NOTEARS

- It can be made non-linear:

$$
\begin{array}{ll}
\text{Find} & A \\
 & g_1, g_2 \ \text{(pointwise)} \\
\text{To minimize} & \text{Mean}_i \left\| X_i - g_2\big(A g_1(X_i)\big) \right\|_F^2 + \lambda \left\| A \right\|_1 \\
\text{Such that} & \text{Trace}\, e^{A \odot A} = d
\end{array}
$$

A graph autoencoder approach to causal structure learning (I. Ng et al., 2019)

Beyond NOTEARS

- It can be made non-linear:

$$
\begin{array}{ll}
\text{Find} & g_W \ \text{neural net} \\
\text{To minimize} & \underset{i}{\text{Mean}} \left\| X - g_W(X) \right\|^2 + \lambda \left\| W \right\|^2 \\
\text{Such that} & \operatorname{Trace} e^{C} = d \\
\text{Where} & C = \left| W^{(L)} \right| \cdot \left| W^{(L-1)} \right| \cdots \left| W^{(1)} \right| \\
 & \text{(neural network connectivity matrix)}
\end{array}
$$

Gradient-based neural DAG learning (S. Lachapelle et al., 2019)

Beyond NOTEARS

- Learn a binary mask (Gumbel softmax)

$$
\begin{array}{ll}
\text{Find} & A \in \{0,1\}^{n \times n} \\
 & g_i : \mathbf{R}^n \to \mathbf{R} \\
 & \\
\text{Such that} & X_i \approx g_i(A_{\cdot i} \odot X) \\
 & A \text{ acyclic}
\end{array}
$$

Masked gradient-based causal structure learning (I. Ng et al., 2022)

Beyond NOTEARS

- The model $X = AX + Z$ defines an auto-encoder

$$X = (I - A)^{-1} Z \quad \text{decoder}$$

$$Z = (I - A) X \quad \text{encoder}$$

- It can be made nonlinear

$$X = f_2\big[(I - A)^{-1} f_1(Z)\big] \quad \text{decoder}$$

$$Z = f_4\big[(I - A) f_3(X)\big] \quad \text{encoder}$$

- And trained as a VAE, with a NOTEARS-like penalty

DAG-GNN: DAG structure learning with graph neural networks (Y. Yu et al.)

Beyond GES

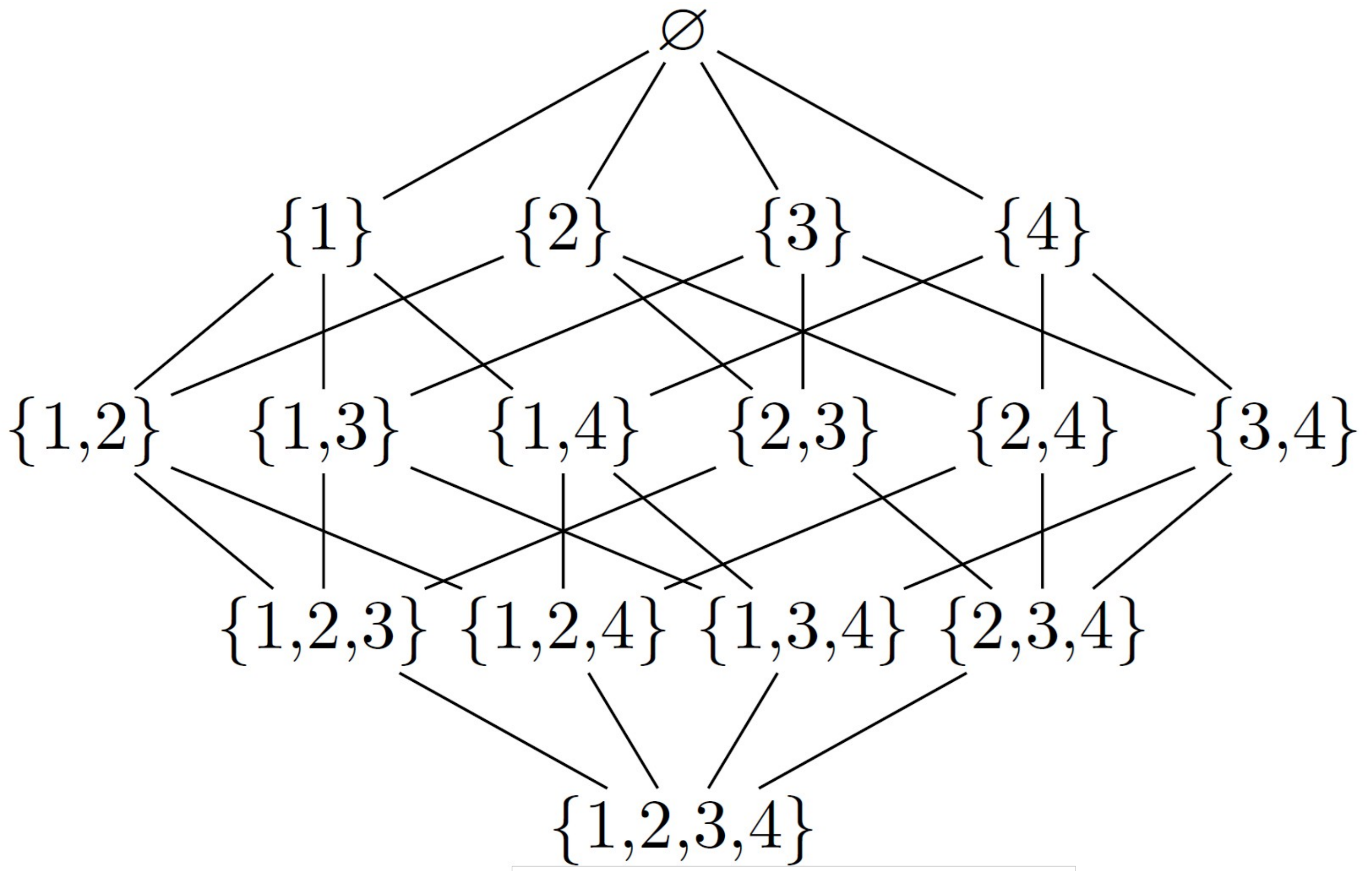
$$\text{Score} = \underset{\substack{j \in [\![1,d]\!] \\ \text{variable}}}{\sum} \; \underset{\substack{k \in [\![1,n]\!] \\ \text{observation}}}{\sum} \log p(x_{jk} | \text{Pa}_{jk}; \theta_j) - \frac{|\theta_j|}{2} \log n$$

Beyond GES

- GES is greedy: it is not an exact search
- Exhaustive search is not reasonable: there are too many DAGs
- It is a shortest path problem on the subset lattice: it can be solved with dynamic programming
- The lasso gives a consistent A* heuristic

A lasso for learning a sparse Bayesian network structure for continuous variables* (J. Xiang and S. Kim, 2013)

Beyond GES



$$2^n \ll n! \ll \#\text{DAGs}$$

A* lasso for learning a sparse Bayesian network structure for continuous variables (J. Xiang and S. Kim, 2013)

Order Search

- First find a topological order, then the DAG

- CAM:
  - Neighbourhood selection (GAM Boosting)
  - Complete DAG (unpenalized GES)
  - Pruning (p-values of GAM terms)

CAM: Causal additive models, high-dimensional order search, and penalized regression (P. Bühlmann et al., 2013)

Reinforcement Learning

- Stochastic search for a high-score DAG:
    - State: DAG
    - Action: new (nearby) DAG
    - Reward: score

Reinforcement Learning

- Stochastic search for a high-score DAG:
  - State: Bootstrap sample
  - Action: DAG
  - Reward: score + $\lambda \cdot 1_{DAG}$ + $\mu \cdot$NOTEARS

Causal discovery with reinforcement learning (S. Zhu et al., 2019)

Reinforcement Learning

- Progressively build a topological order:
  - State: embedding of the latest variable selected
  - Action: variable to add
  - Reward: BIC improvement
  - Encoder: self-attention $R^{n \times d} \rightarrow R^{n \times d}$
  - Decoder: LSTM
  - Optimization: actor critic, policy gradient

Causal discovery with reinforcement learning (S. Zhu et al., 2019)

LLM

LLM

- Ask an LLM to give the causal graph, from **metadata** alone (column names + descriptions)
- Ask an LLM to scour the literature to extract causal relations, and put them in a database (knowledge graph).

Efficient Causal Graph Discovery Using Large Language Models (T. Jiralerspong et al., 2024)

LLM

- Ask an LLM to give the causal graph, from **metadata** alone (column names + descriptions)
- Ask an LLM to scour the literature to extract causal relations, and put them in a database (knowledge graph).

Code

gCastle

```
import castle.algorithms
import networkx as nx

model = castle.algorithms.PC()
model.learn(X)

A = model.causal_matrix
A = pd.DataFrame( A, columns = X.columns, index = X.columns )
g = nx.from_pandas_adjacency( A, create_using = nx.DiGraph )
```

causal-learn

```python
from causallearn.search.ConstraintBased.PC import pc
from causallearn.utils.cit import fisherz, kci, chisq, gsq

# Computation
g = pc(X.values)

# Extract the adjacency matrix
A = pd.DataFrame( g.G.graph )
A = ( A == -1 ).astype(int)
A.columns = A.index = X.columns.copy()
```

causal-learn

```python
from causallearn.search.ScoreBased.GES import ges

r = ges(X.values)

A = pd.DataFrame( r['G'].graph )
A = ( A == -1 ).astype(int)
A.columns = A.index = X.columns.copy()
```

causal-learn

```
from causallearn.search.FCMBased.lingam import ICALiNGAM

model = ICALiNGAM()
model.fit(X)

A = model.adjacency_matrix_
A = pd.DataFrame( A != 0, index = X.columns, columns = X.columns )
```

cdt

```
import cdt                              # Causal discovery toolbox
import networkx as nx

pc = cdt.causality.graph.PC()           # Continuous, Gaussian variables
g = pc.predict(d)

A = nx.adjacency_matrix(g).todense()
A = pd.DataFrame( X, index = g.nodes, columns = g.nodes )
```

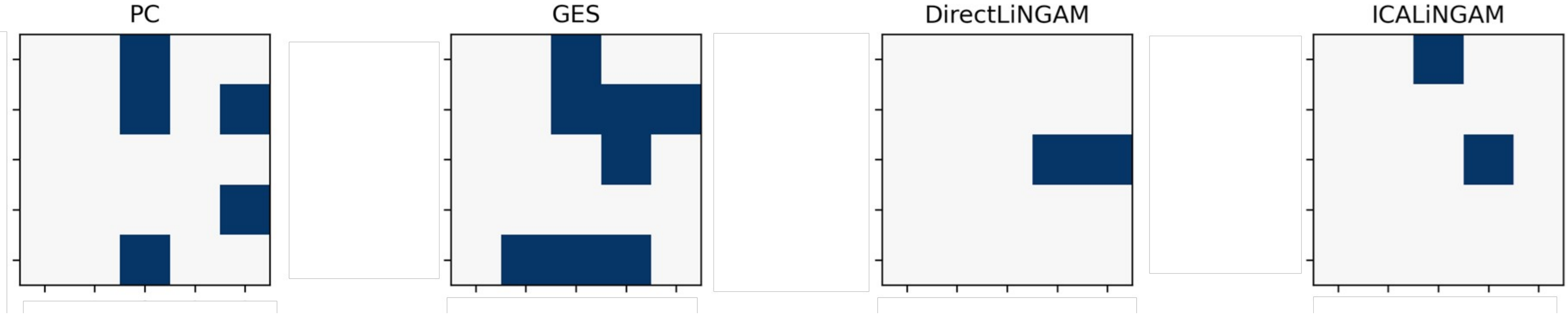
dowhy

```
from dowhy import CausalModel
model = CausalModel(
    data      = X,
    treatment = "T",
    outcome   = "Y",
    graph     = ' '.join( nx.generate_gml(g) ),
)
estimand = model.identify_effect()
estimate = model.estimate_effect(
    estimand,
    method_name = "backdoor.linear_regression",
)
model.refute_estimate(
    estimand, estimate,
    method_name = "random_common_cause",
)
```

Examples

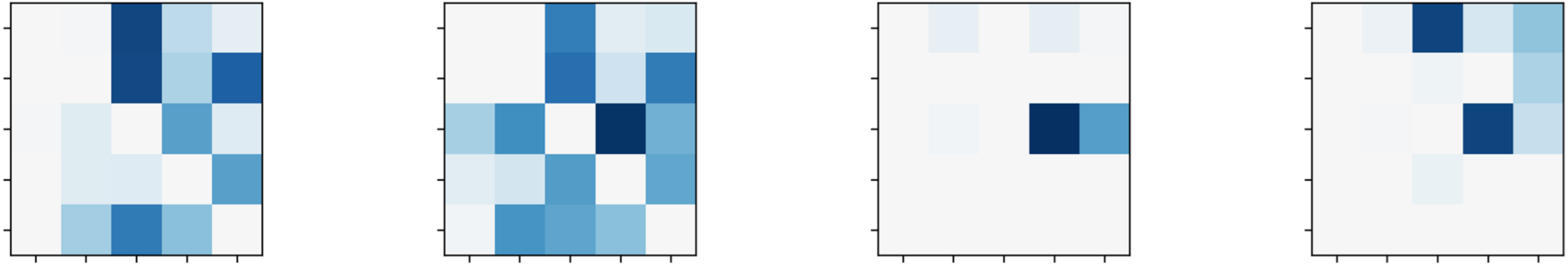
Examples

- Different algorithms give very different results



Examples

- Different bootstrap samples give very different results



Conclusion

Summary

- **PC**: conditional independence test
- **GES**: goodness-of-fit of models predicting y from its parents
- **LiNGAM**: independent component analysis
- **NOTEARS**: optimization problem, with acyclicity constraint

- Software: gCastle, causal-learn, cdt

Conclusion

- Do not start from data, but from a domain knowledge causal graph
- Do not use a single causal discovery algorithm, but several
- Do not run them on just the data, but also on bootstrap samples
- Do not only look at the output, look at the ingredients of the algorithms

References

Introduction to Causal Inference (B. Neal, 2020)

A survey on causal discovery: theory and practice (A. Zanga and F. Stella, 2023)

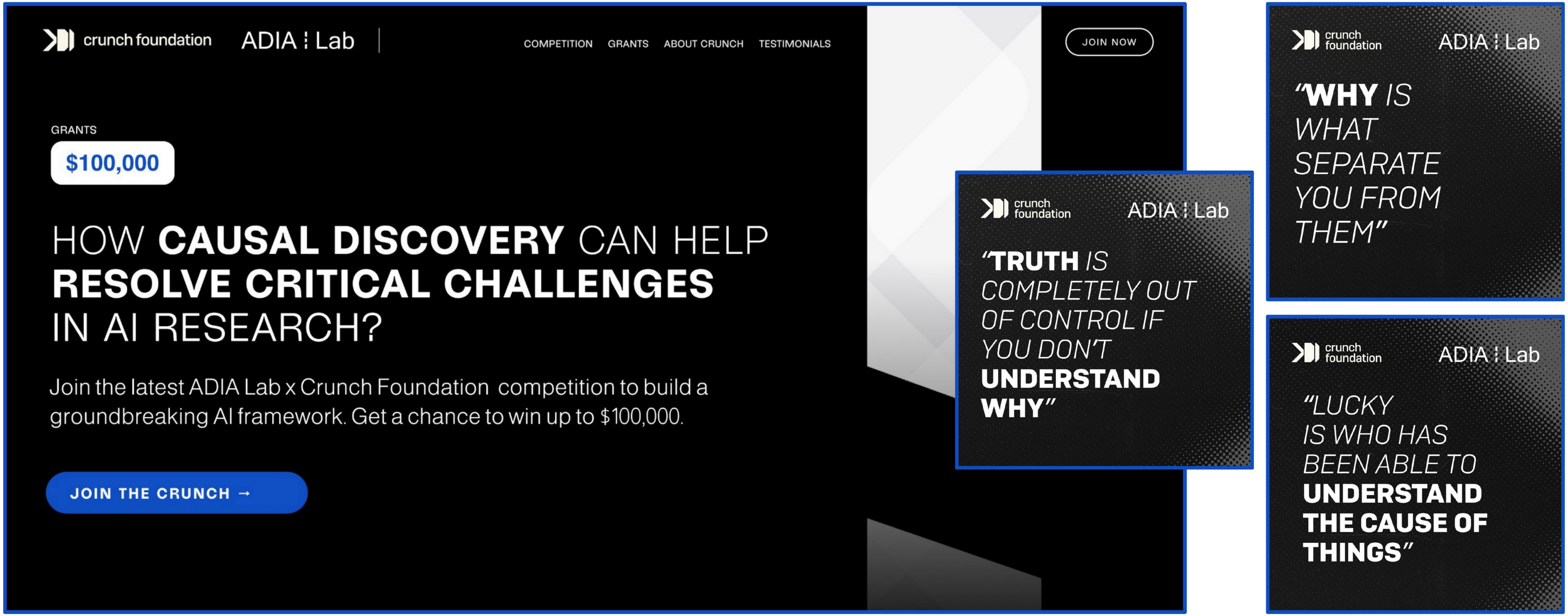
https://github.com/huawei-noah/trustworthyAI/tree/master/gcastle

https://causal-learn.readthedocs.io/en/latest/

https://cran.r-project.org/web/views/CausalInference.html#dag

Extra Slides

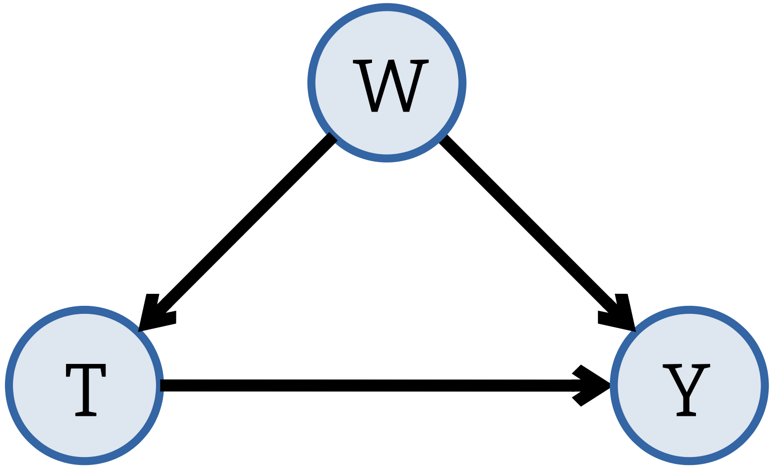
Causality competition



Backdoor adjustment

$$\mathrm{E}\left[Y|\mathrm{do}(T=t)\right] = \mathrm{E}_W \mathrm{E}\left[Y|T=t, W\right]$$

$$\mathrm{P}\left[Y|\mathrm{do}(T=t)\right] = \sum_w \mathrm{P}\left[Y|t,w\right]\mathrm{P}[w]$$



With linear models: fit a regression

$$Y = \alpha + \beta\, T + \gamma\, W$$

and average W out:

$$Y = \alpha + \beta\, T + \gamma\, \mathrm{E}[W]$$

Kernel methods

- Many machine learning algorithms do not really require coordinates, but just the Gram matrix $K_{ij}=\langle x_i,x_j\rangle$.
- Increasing the dimension, $K_{ij}=\langle \phi(x_i), \phi(x_i)\rangle$ does not change the size of the Gram matrix.
- We do not even need to compute $\phi(x_i)$: we just need a (positive definite) kernel, $K(x,y)=\langle \phi(x), \phi(y)\rangle$.

BIC Score

$$\text{Score} = \sum_{\substack{j \in [\![1,d]\!] \\ \text{variable}}} \sum_{\substack{k \in [\![1,n]\!] \\ \text{observation}}} \log p(x_{jk} | \text{Pa}_{jk}; \theta_j) - \frac{|\theta_j|}{2} \log n$$

Local BIC score of $X \to y$

$H = \log(\Sigma[y,y] - \Sigma[y,X]\, \Sigma[X,X]^{-1}\, \Sigma[X,y])$

$-BIC = n \cdot H + \lambda \cdot k \cdot \log(n)$

n = number of observations

k = number of variables in X

Σ = variance matrix of the data

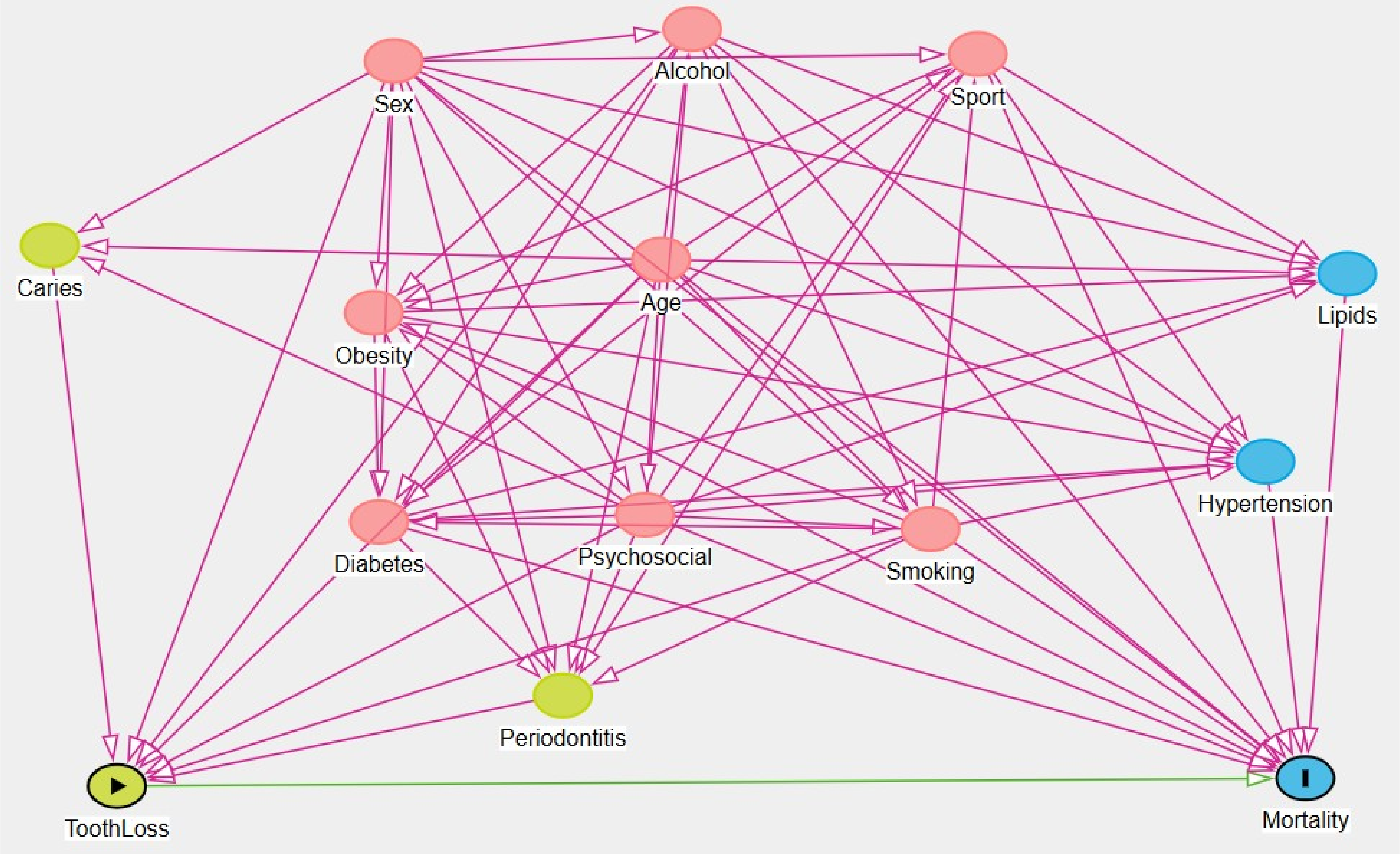
Markov equivalence class

A **CPDAG** (complete partially directed acyclic graph) is a PDAG where

- All undirected edges are **reversible** (you can choose either direction, that does not change the MEC)
- All directed edges are **compelled** (if you flip them, the graph moves to another MEC)

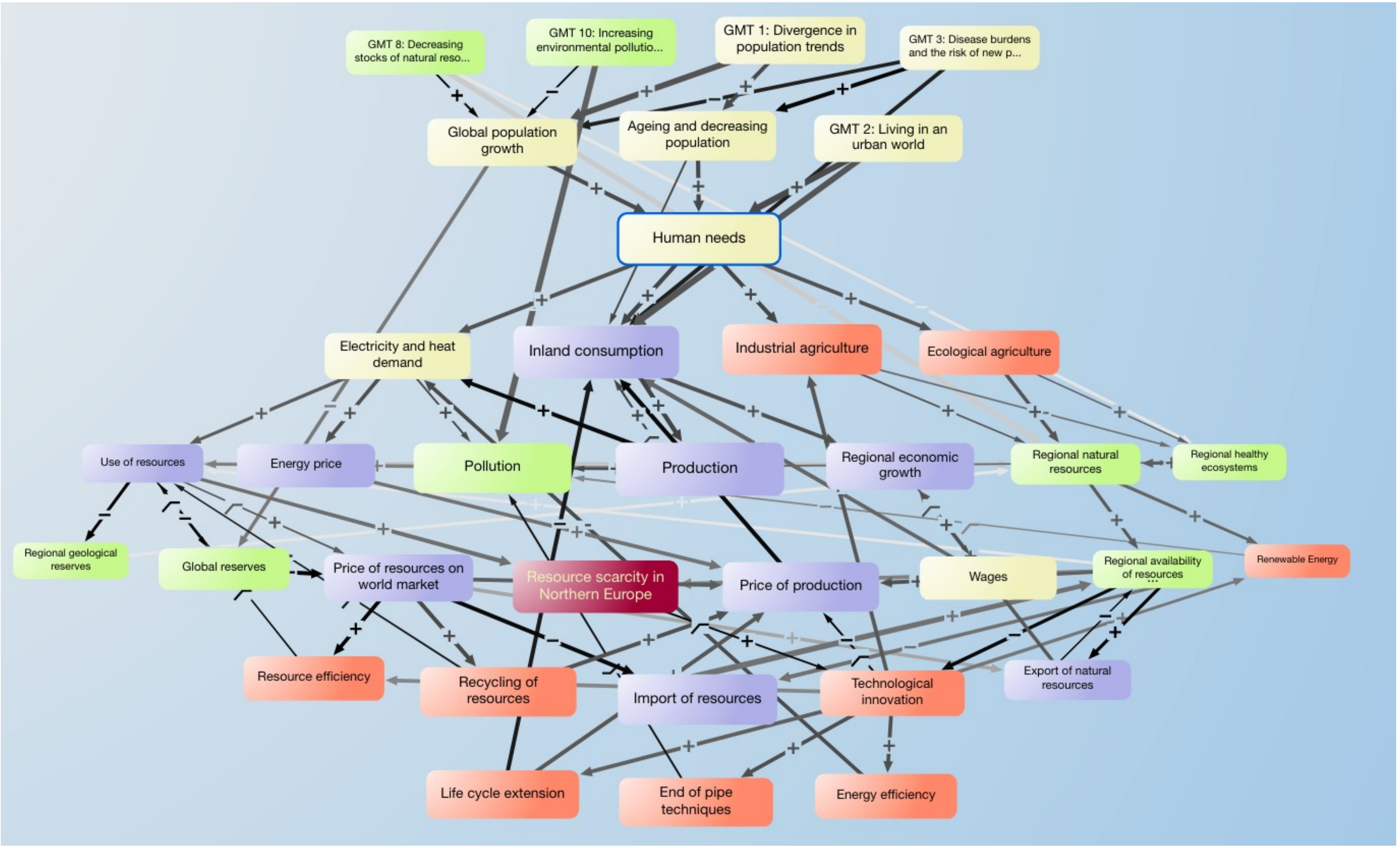
Unused Slides

Real-world causal graphs



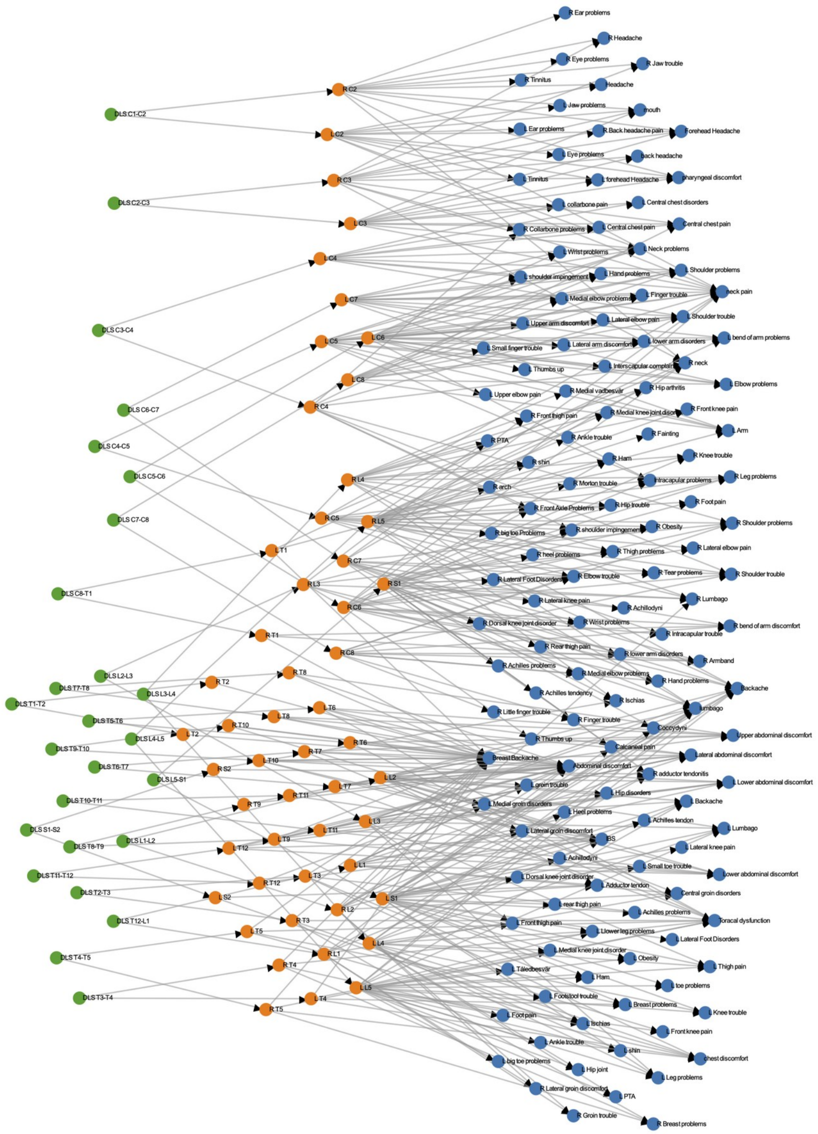
https://www.dagitty.net/dags.html

Real-world causal graphs



Impact assessment of global megatrends (U. Lorenz and H.V. Haraldson, 2014)

Real-world causal graphs

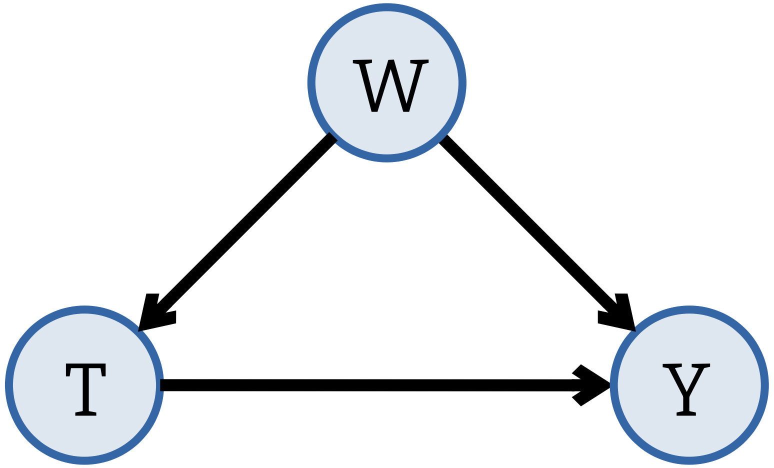


https://arxiv.org/abs/1906.01732

Variants of those algorithms

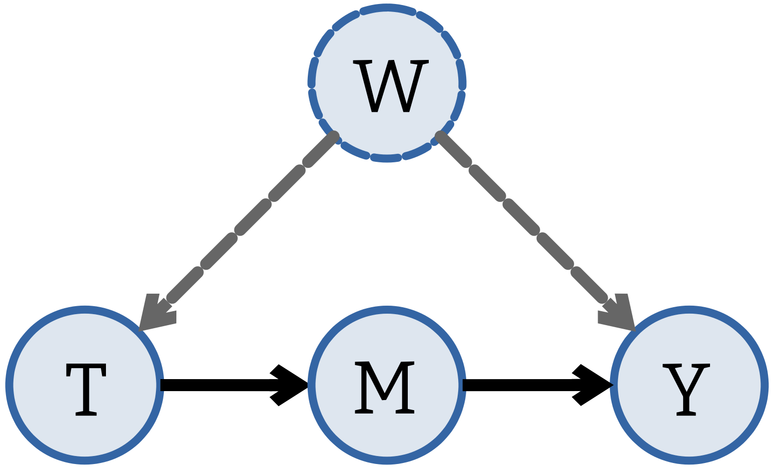
- FCI: PC variant, allowing for unobserved confounders
- FGES: faster implementation of GES
- ARGES: another GES variant, for high-dimensional data
- GFCI: GES variant allowing for unobserved confounders (FCI on the FGES skeleton)
- CCD: PC/FCI with feedback (cycles)
- LiNG: LiNGAM with feedback
- Other LiNGAM variants: non-linear, post-non-linear
- CD-NOD

Back-door adjustment



$$P[Y|\mathrm{do}(T=t)] = \sum_w P[Y,t,w]P[w]$$

Front-door adjustment

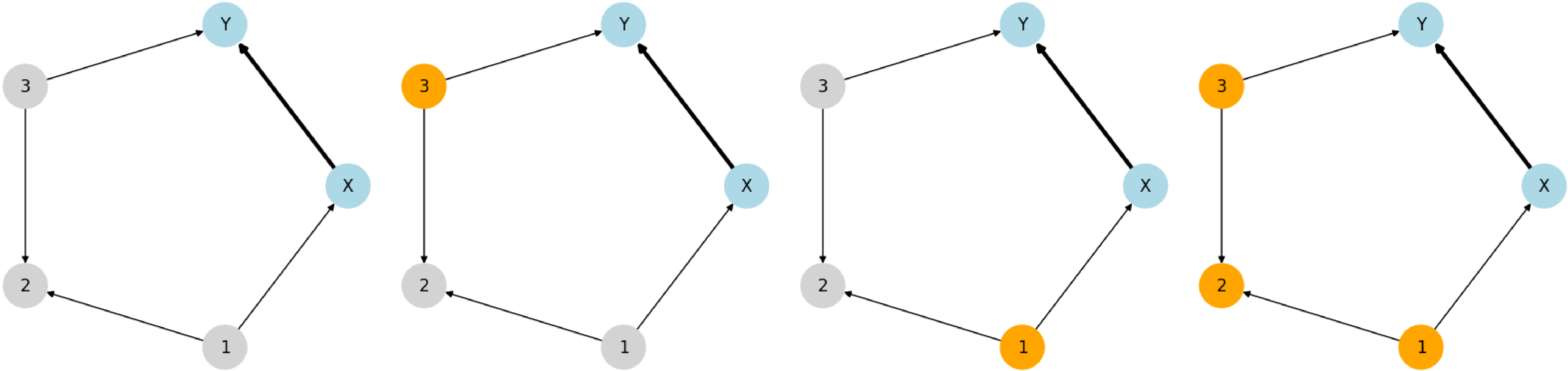


$$P[Y|\mathrm{do}(t)] = \sum_m P[m|t] \sum_t P[y|m,t']P[t']$$

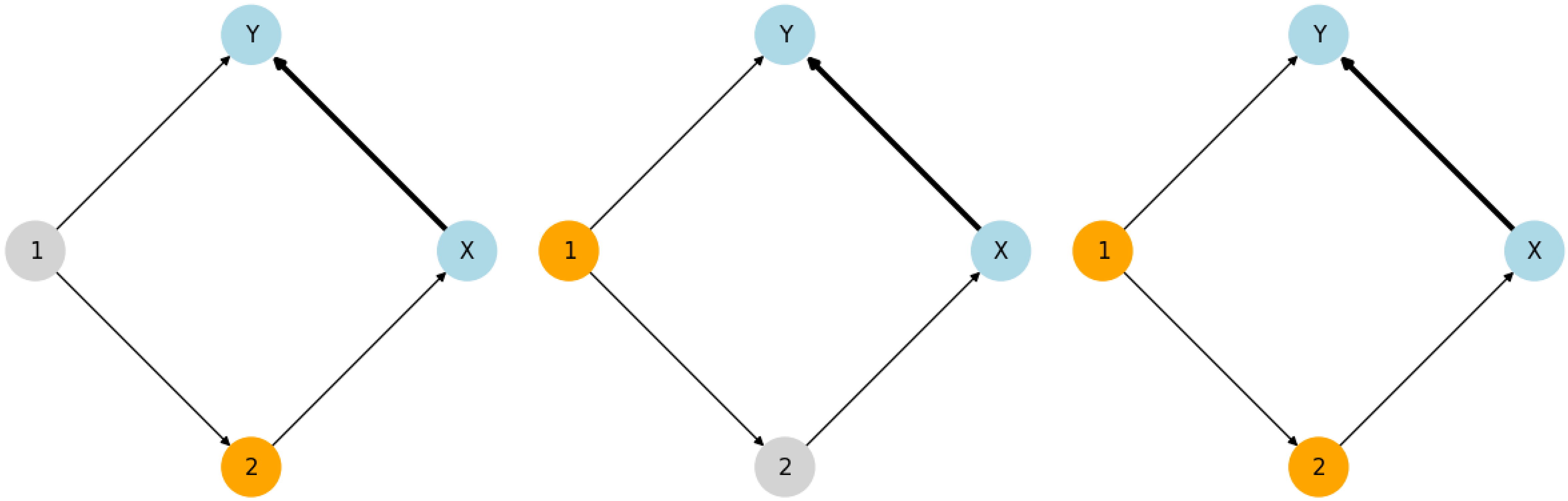
Causal inference

- The sufficient conditioning set is not unique.
- The parents of X form a sufficient conditioning set, but it may be needlessly large.
- More generally, a sufficient conditioning set is a set of nodes blocking all the non-causal paths from X to Y, and leaving all the causal paths open.
- If some variables are not observed, things get more complicated, but do-calculus can tell you if the effect can be estimated from observational data alone.

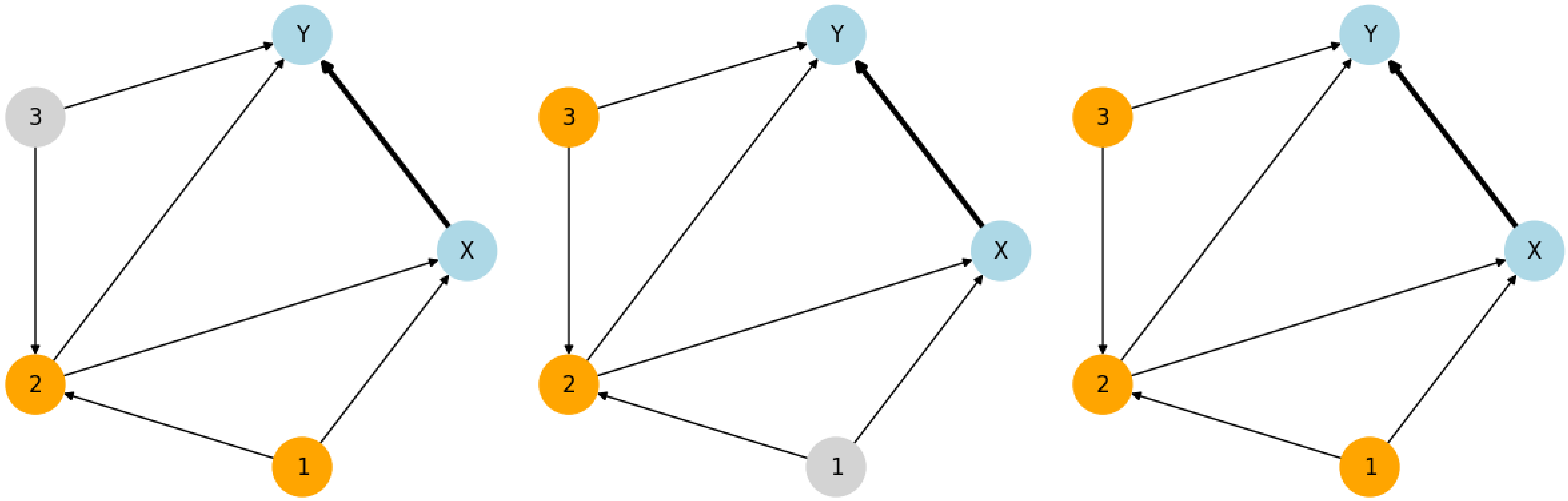
Sufficient Conditioning Sets

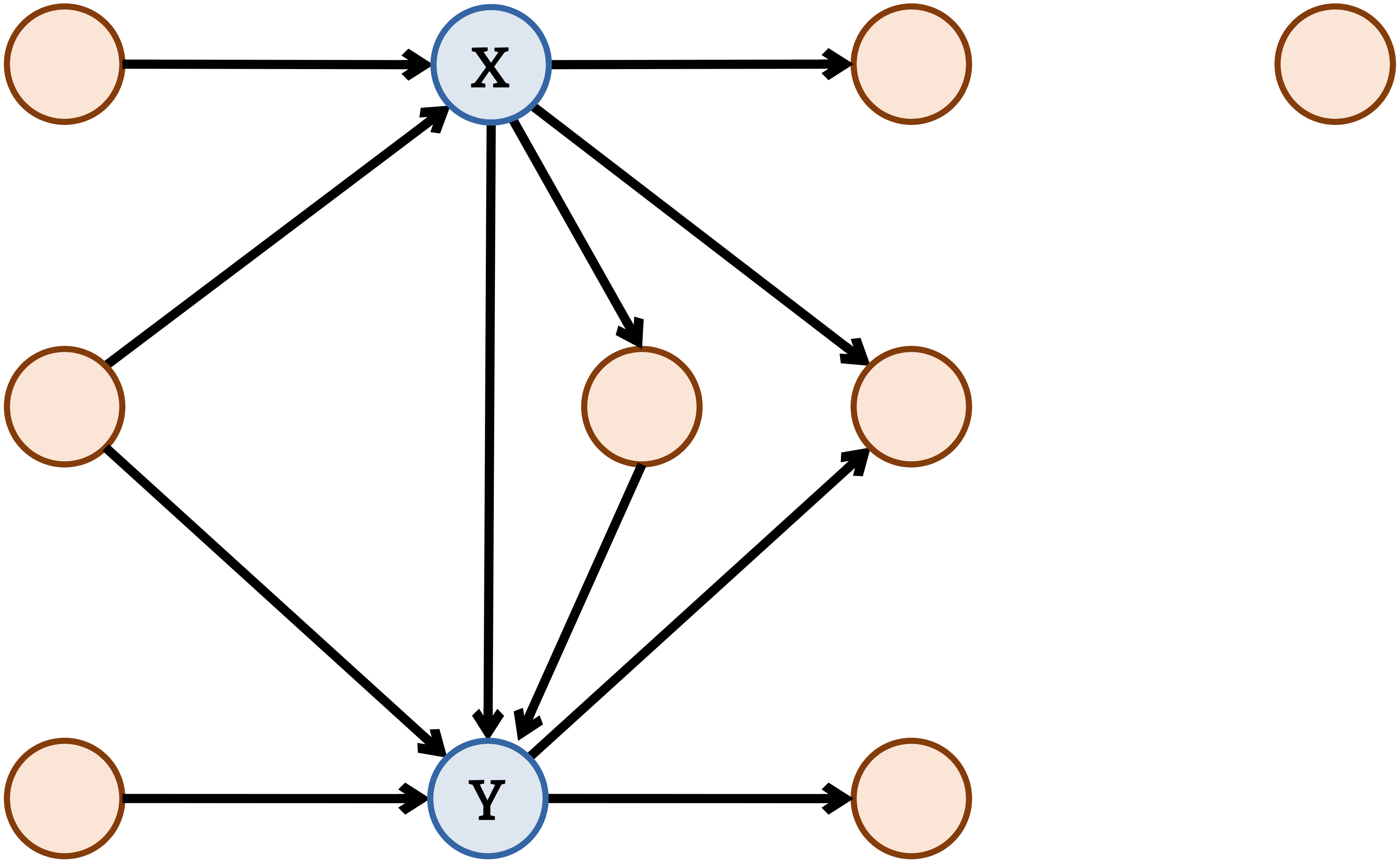


Sufficient Conditioning Sets

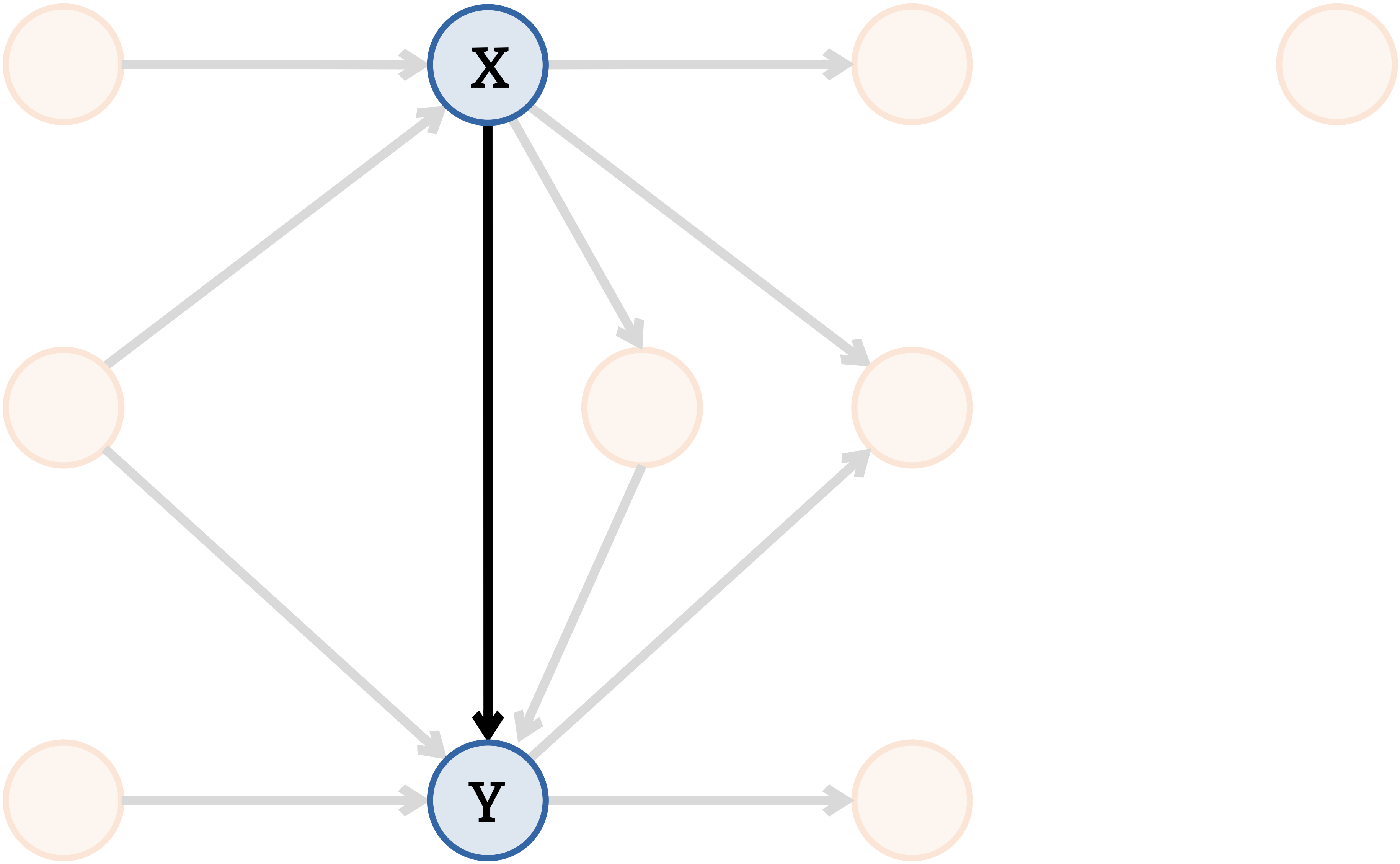


Sufficient Conditioning Sets

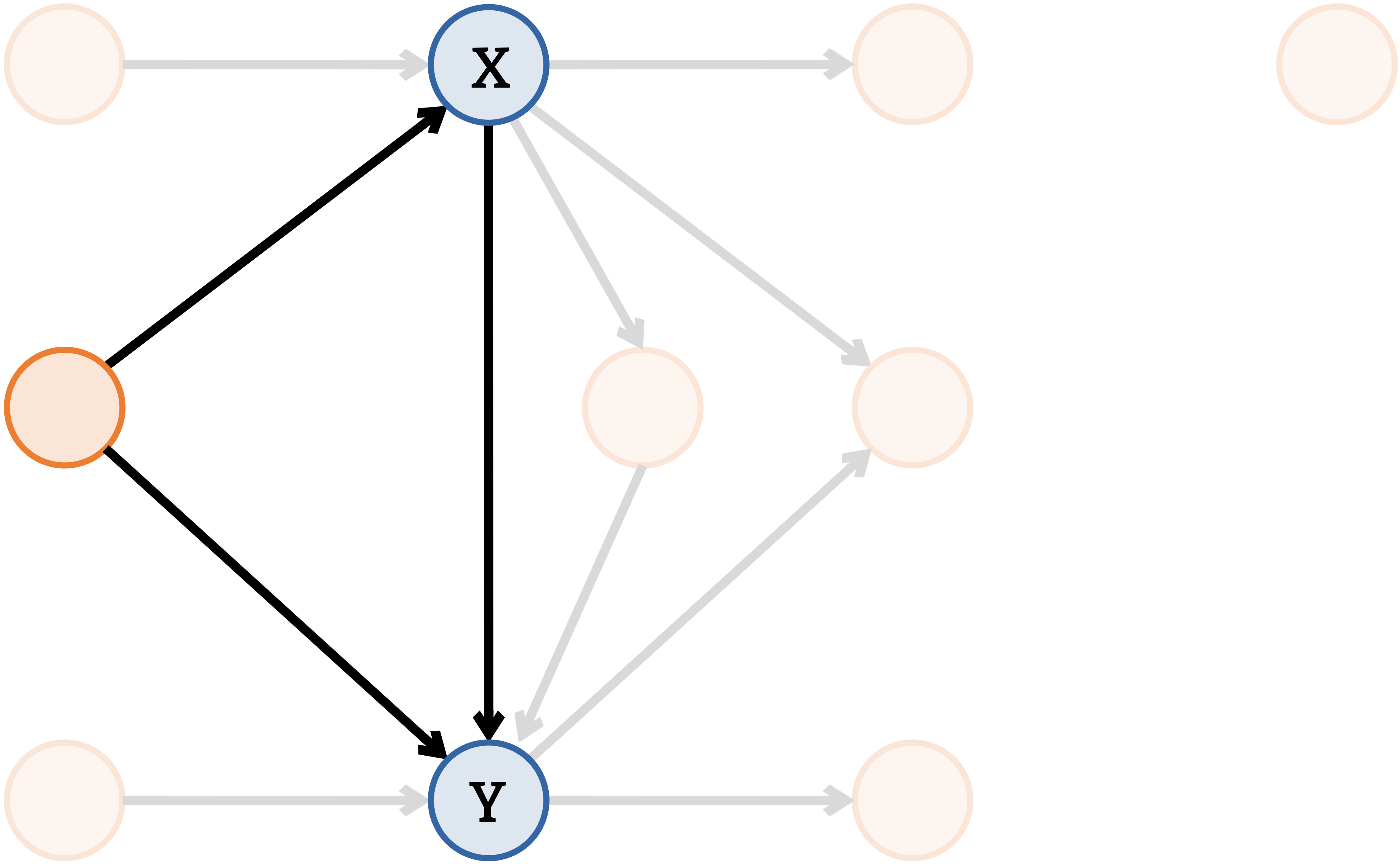



X
Y

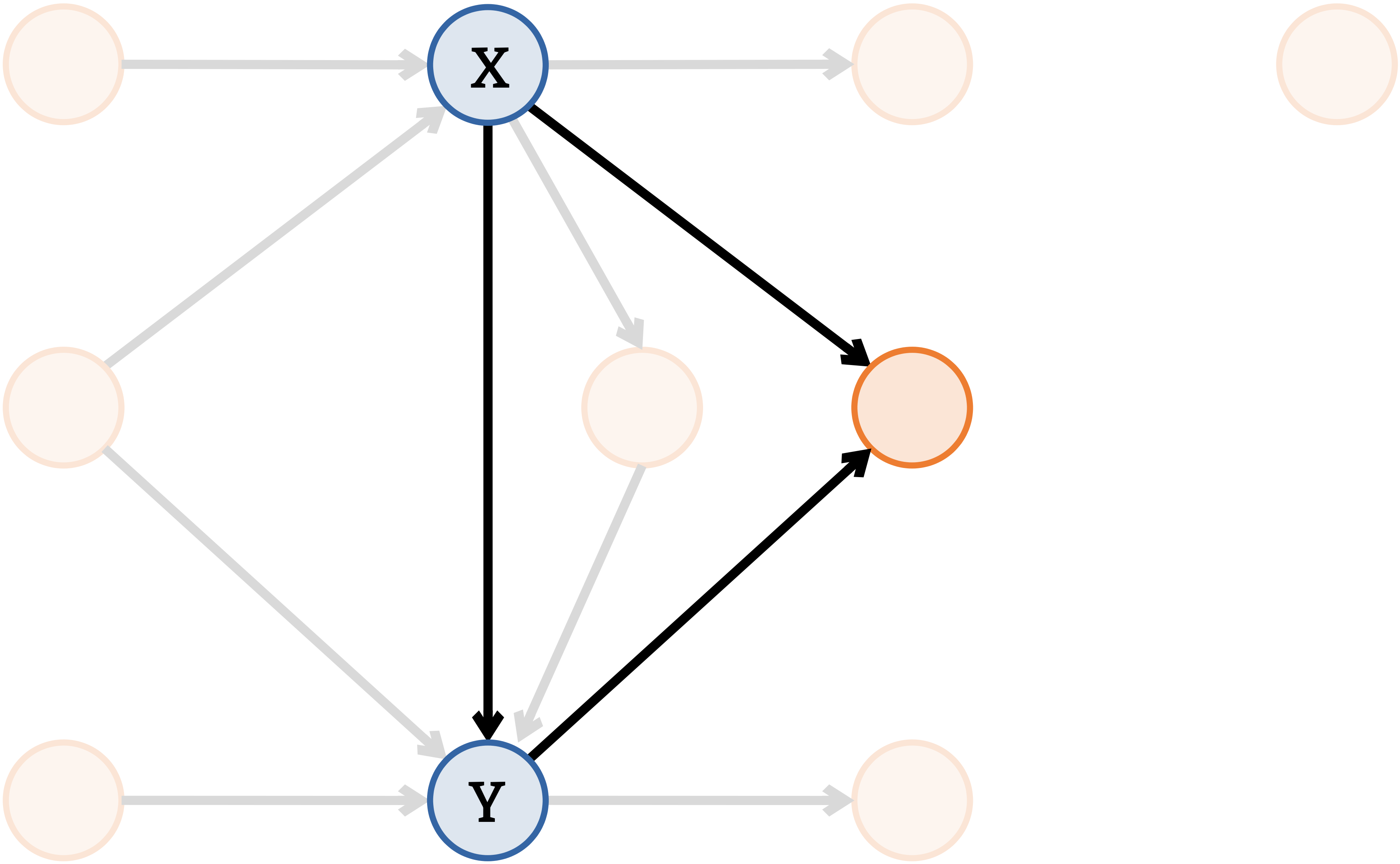
Relation of interest



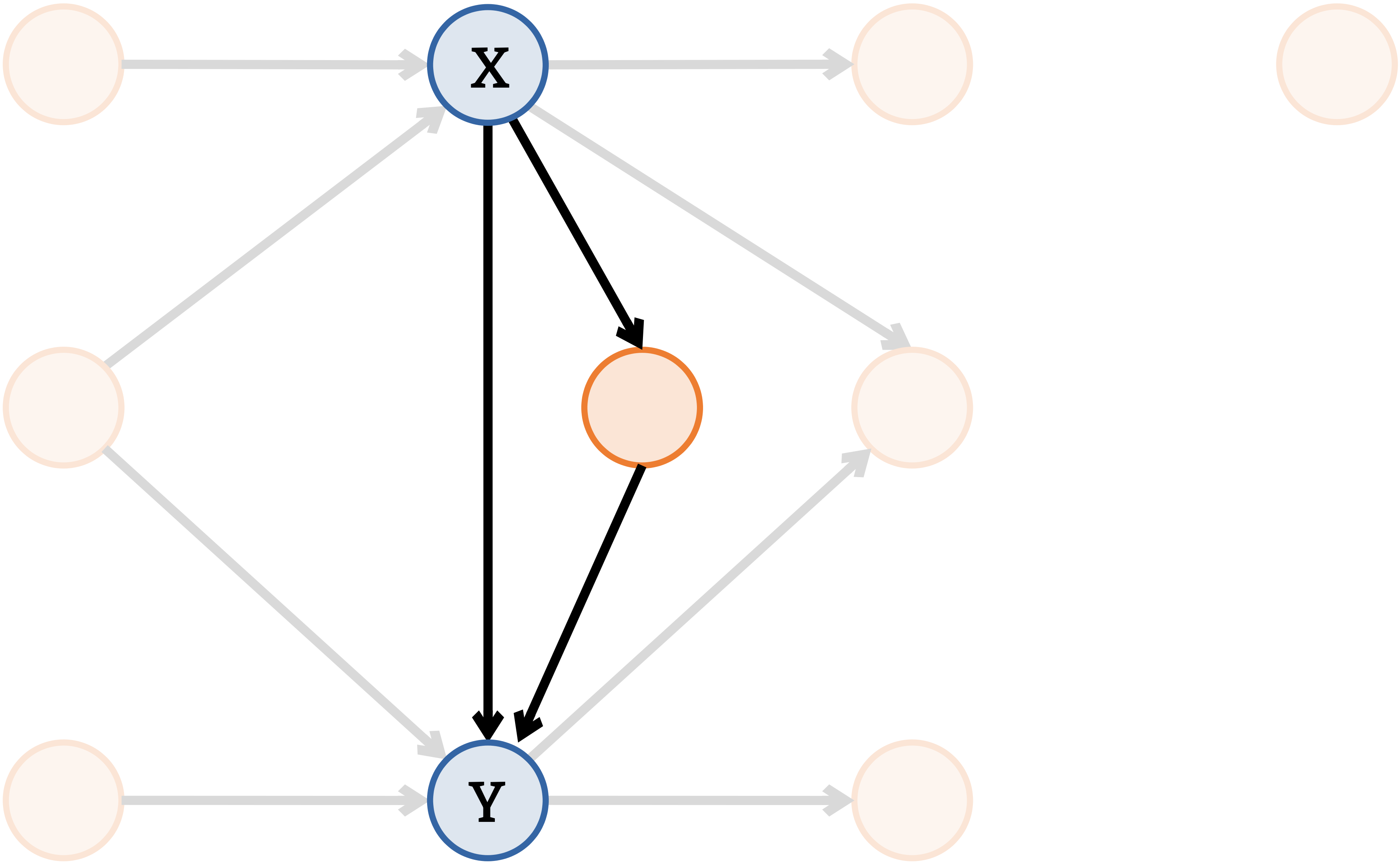
Confounder



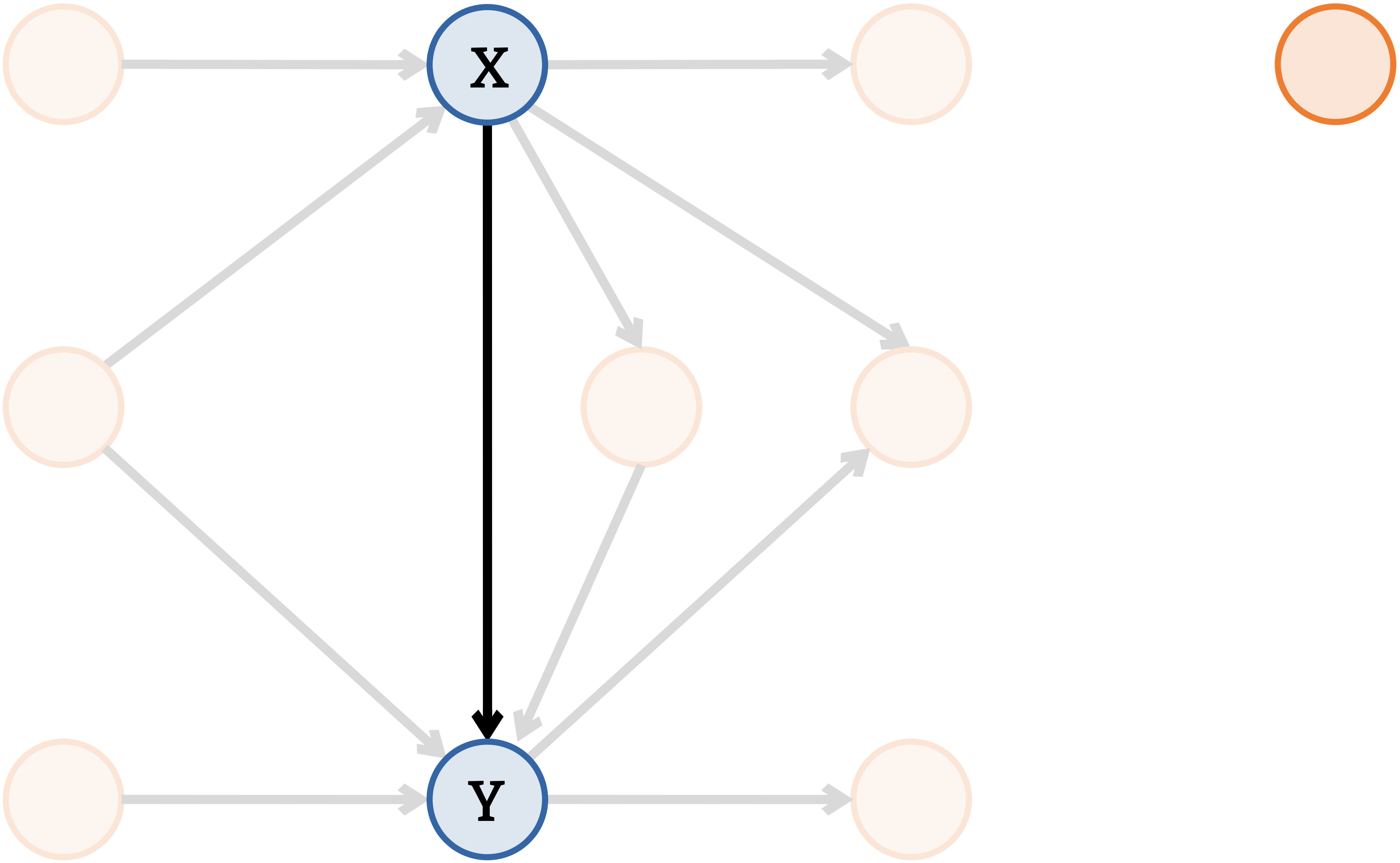
Collider



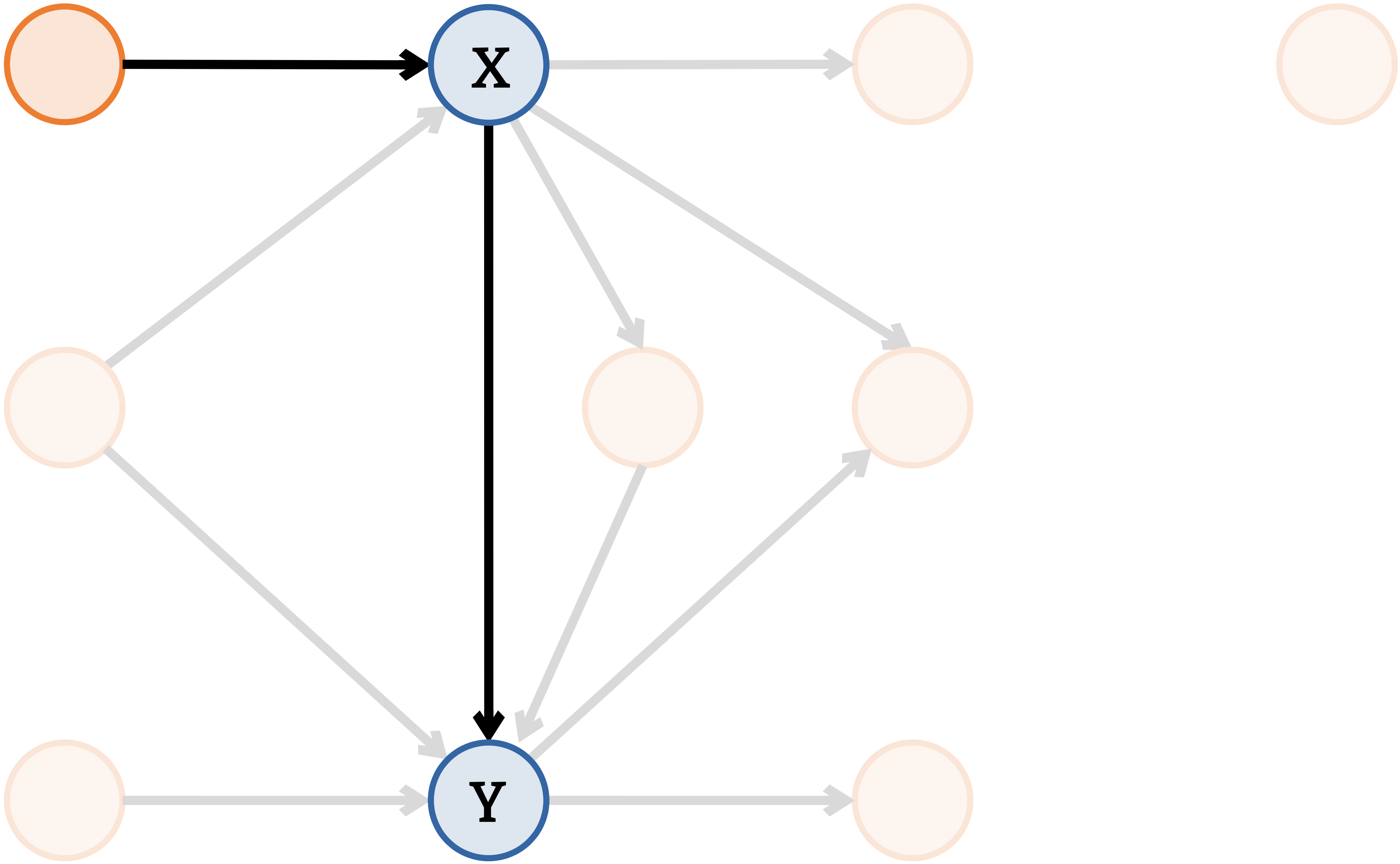
Mediator



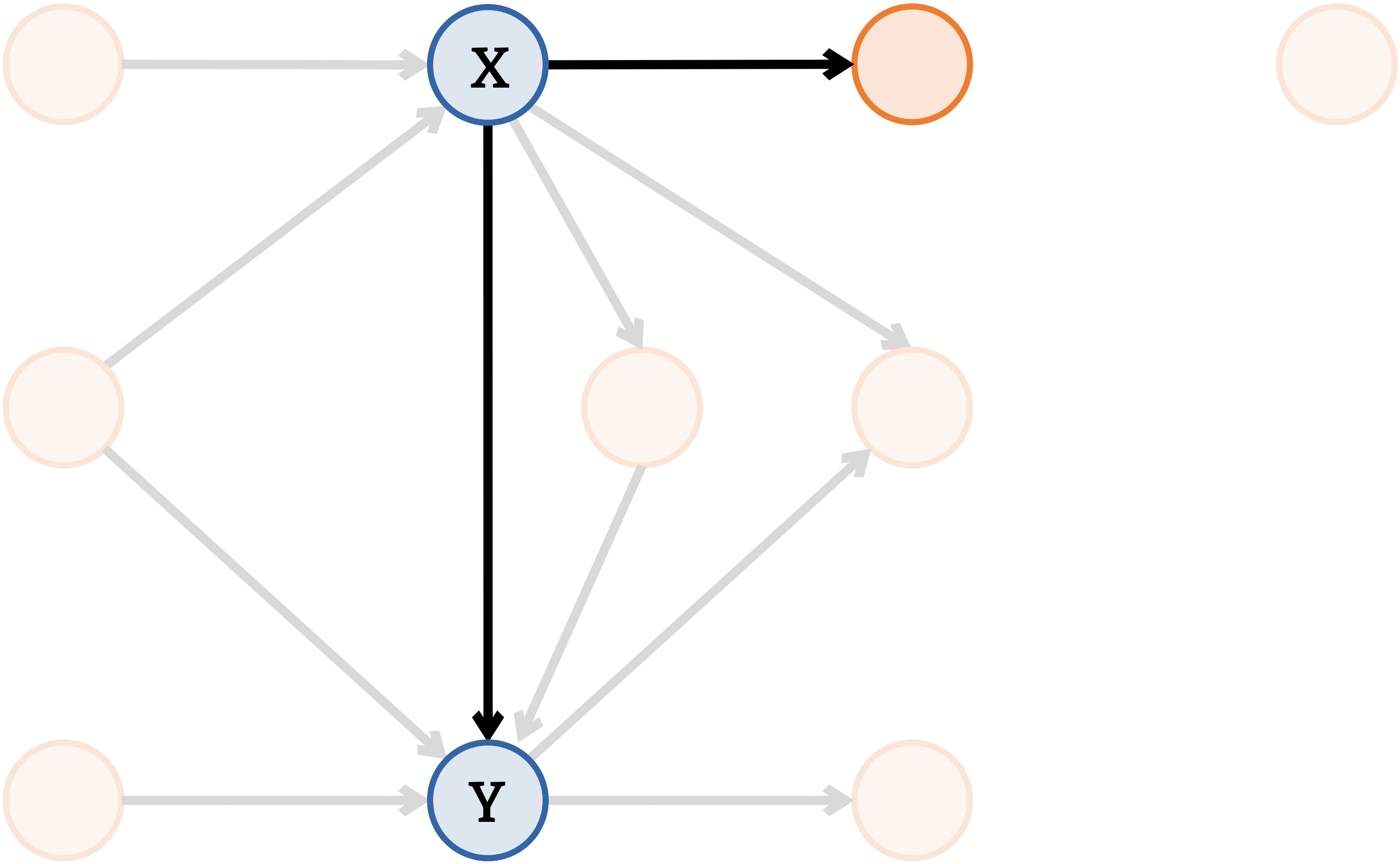
Independent



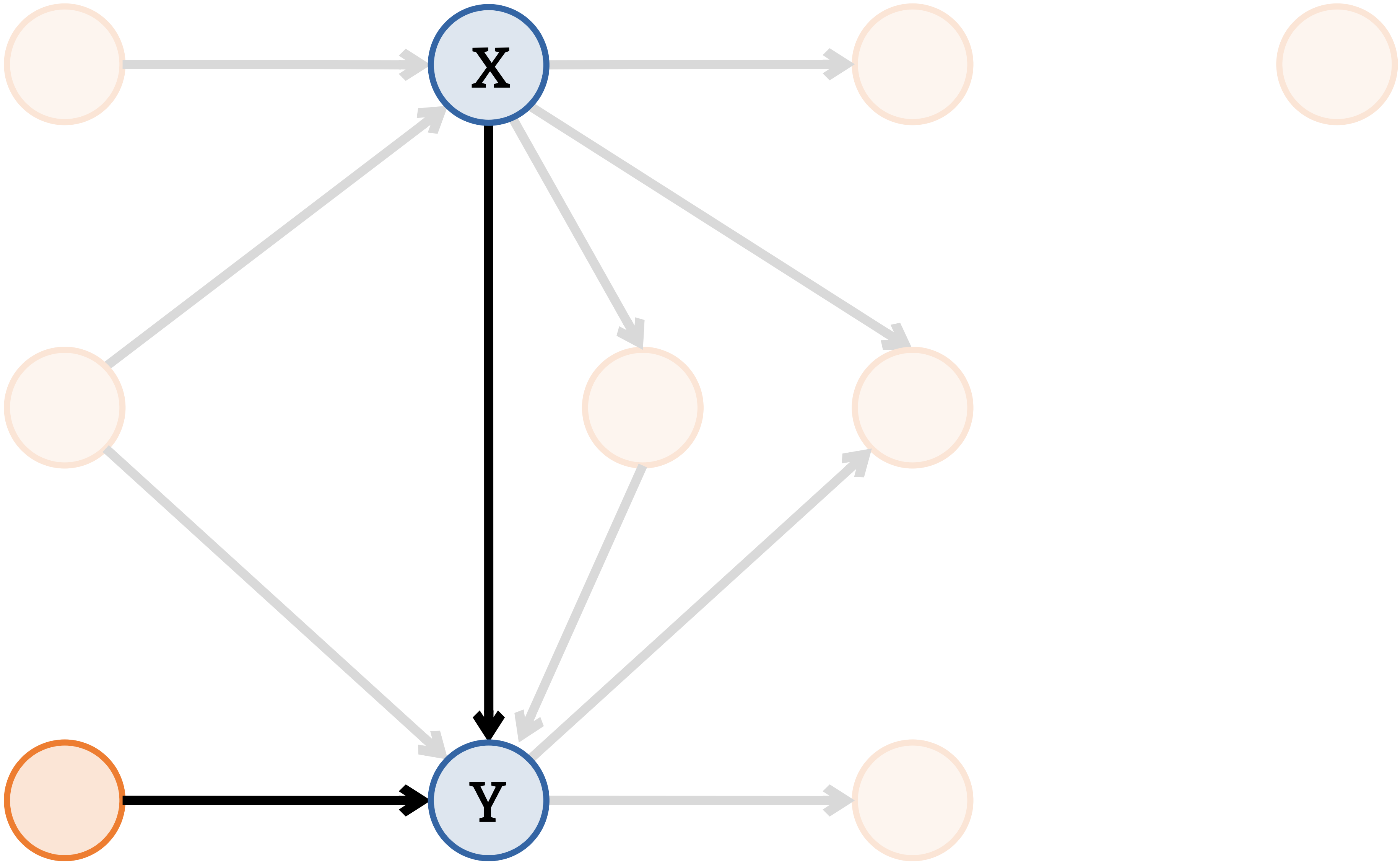
Cause of X



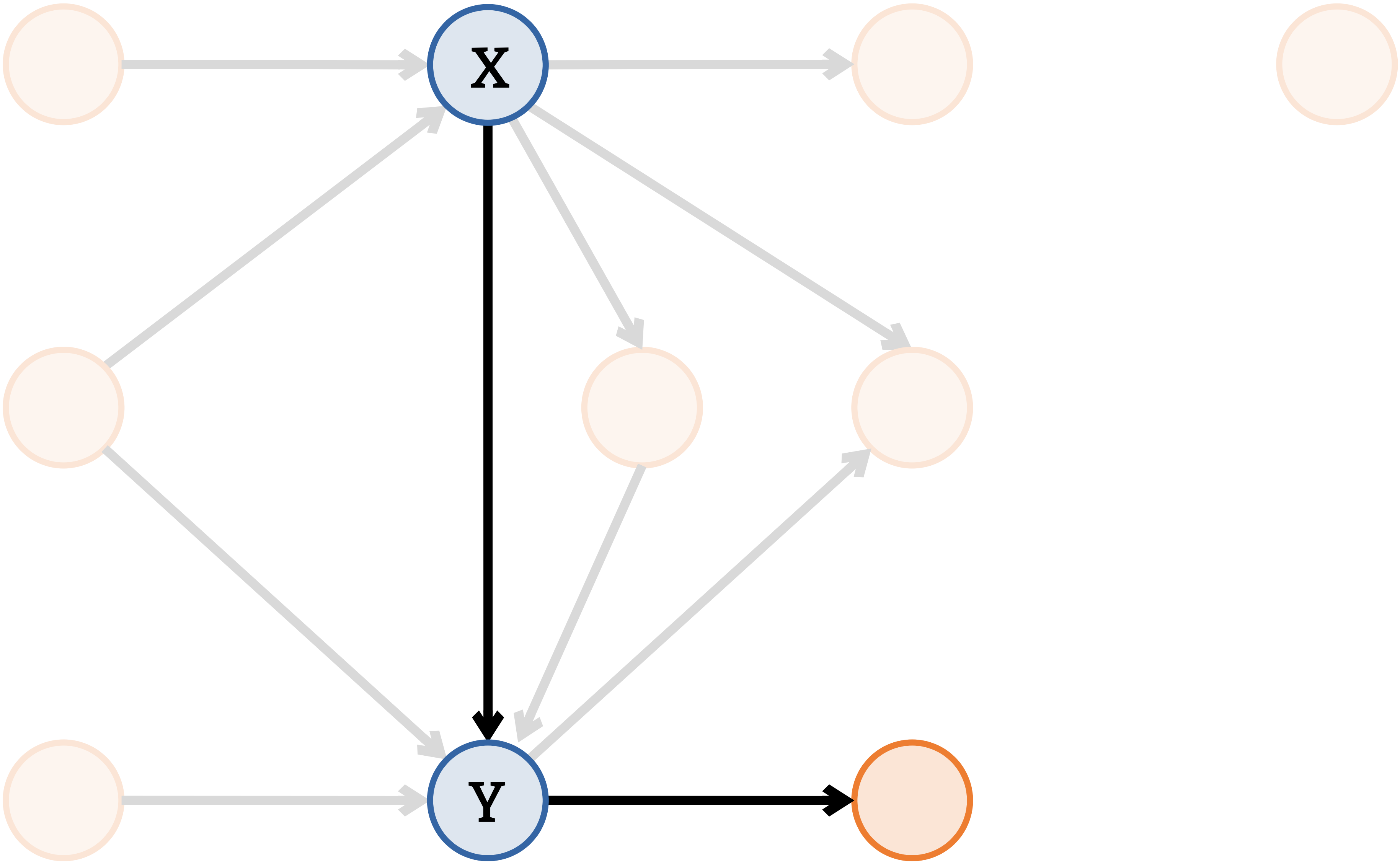
Consequence of X



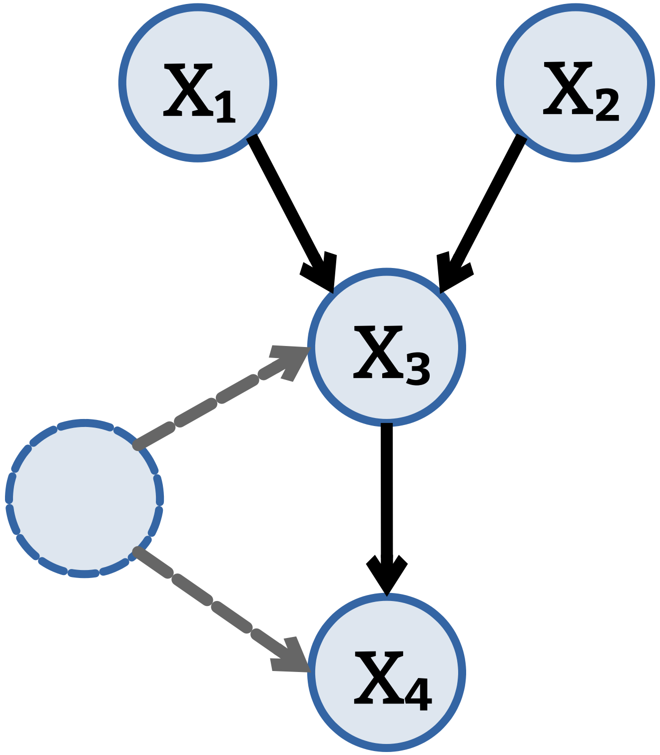
Cause of Y



Consequence of Y

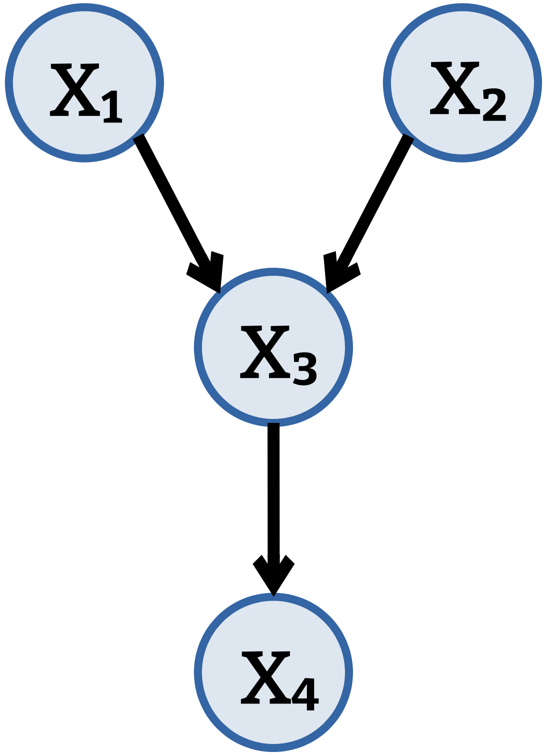


Unobserved confounder



$$X_1 \not\perp X_4 \mid X_3$$

Unobserved confounder



$$X_1 \perp X_4 \mid X_3$$

Beyond NOTEARS

- **NOTEARS is linear:**

$$\begin{array}{ll} \text{Find} & A \\ \text{To minimize} & \text{Mean}_i \left\| X_i - AX_i \right\|_F^2 + \lambda \left\| A \right\|_1 \\ \text{Such that} & \text{Trace}\, e^{A \odot A} = d \end{array}$$

- **It can be made non-linear:**

$$\begin{array}{ll} \text{Find} & A \\ & g_1, g_2 \ \text{(pointwise)} \\ \text{To minimize} & \text{Mean}_i \left\| X_i - g_2\big(A g_1(X_i)\big) \right\|_F^2 + \lambda \left\| A \right\|_1 \\ \text{Such that} & \text{Trace}\, e^{A \odot A} = d \end{array}$$

A graph autoencoder approach to causal structure learning (I. Ng et al., 2019)