Causal Inference - Part B

Gloria Beraldo (gloria.beraldo@unipd.it)
Department of Information Engineering, University of Padova

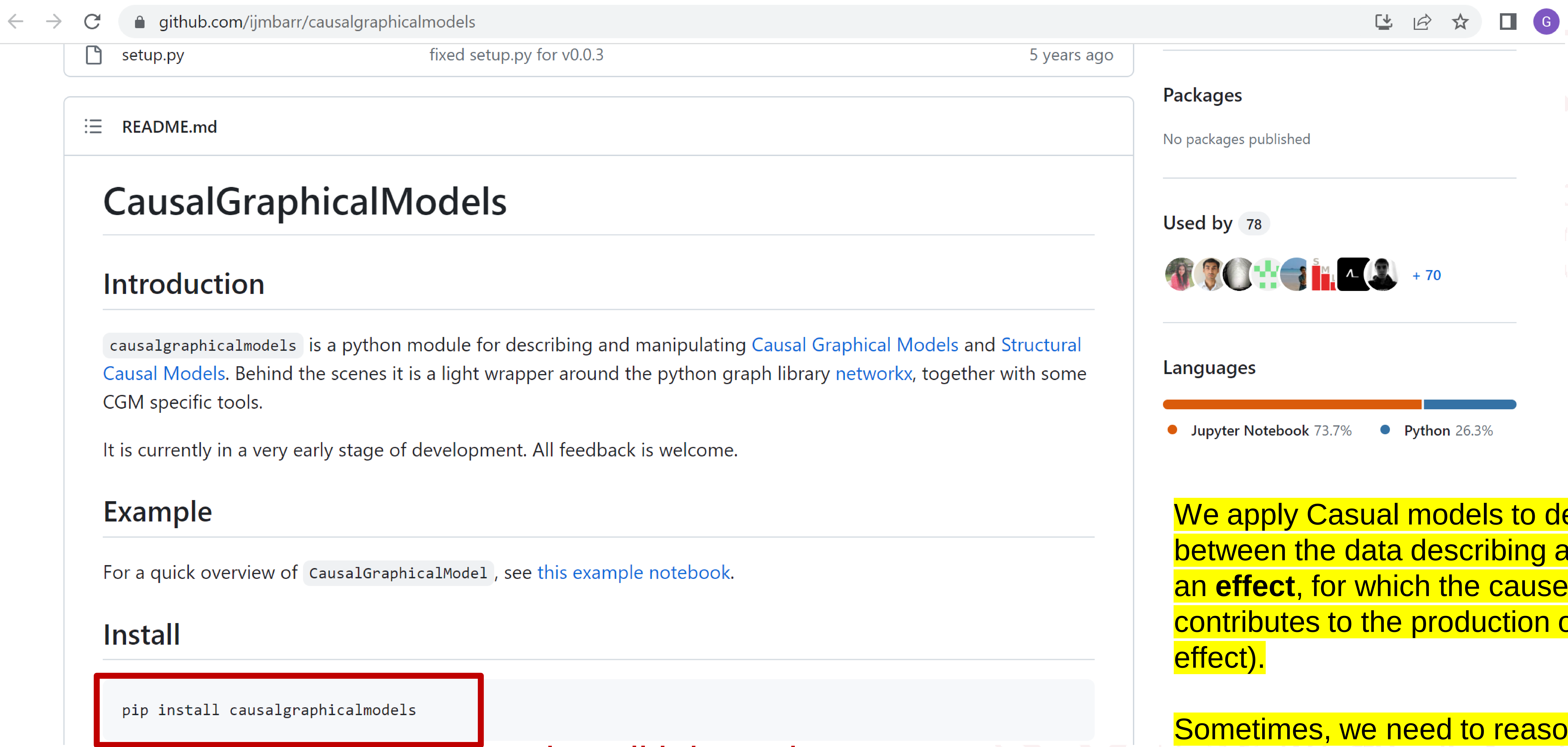
Topics:

- CausalGraphicalModels library
- Application of CausalGraphicalModels to the Sprinkler example
- Causal Inference with Causal Graphical Models
- Recap on adjustment formula
- Adjustment formula & Backdoor criterion
- Adjustment formula & Backdoor criterion: Example
- Recap on front-door adjustment
- Unobserved confounders: Smoking example

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

CausalGraphicalModels library

In this laboratory, we will use another easy-to-use library to implement Causal Graphical Models and in particular Structural Causal Models, including causal networks.



github.com/ijmbarr/causalgraphicalmodels

setup.py fixed setup.py for v0.0.3 5 years ago

README.md

CausalGraphicalModels

Introduction

`causalgraphicalmodels` is a python module for describing and manipulating Causal Graphical Models and Structural Causal Models. Behind the scenes it is a light wrapper around the python graph library networkx, together with some CGM specific tools.

It is currently in a very early stage of development. All feedback is welcome.

Example

For a quick overview of `CausalGraphicalModel`, see this example notebook.

Install

```
pip install causalgraphicalmodels
```

Packages

No packages published

Used by 78

+ 70

Languages

Jupyter Notebook 73.7% Python 26.3%

To install it in Python

We apply Casual models to describe relationships between the data describing a **cause** and an **effect**, for which the cause is an event that contributes to the production of another event, the effect).

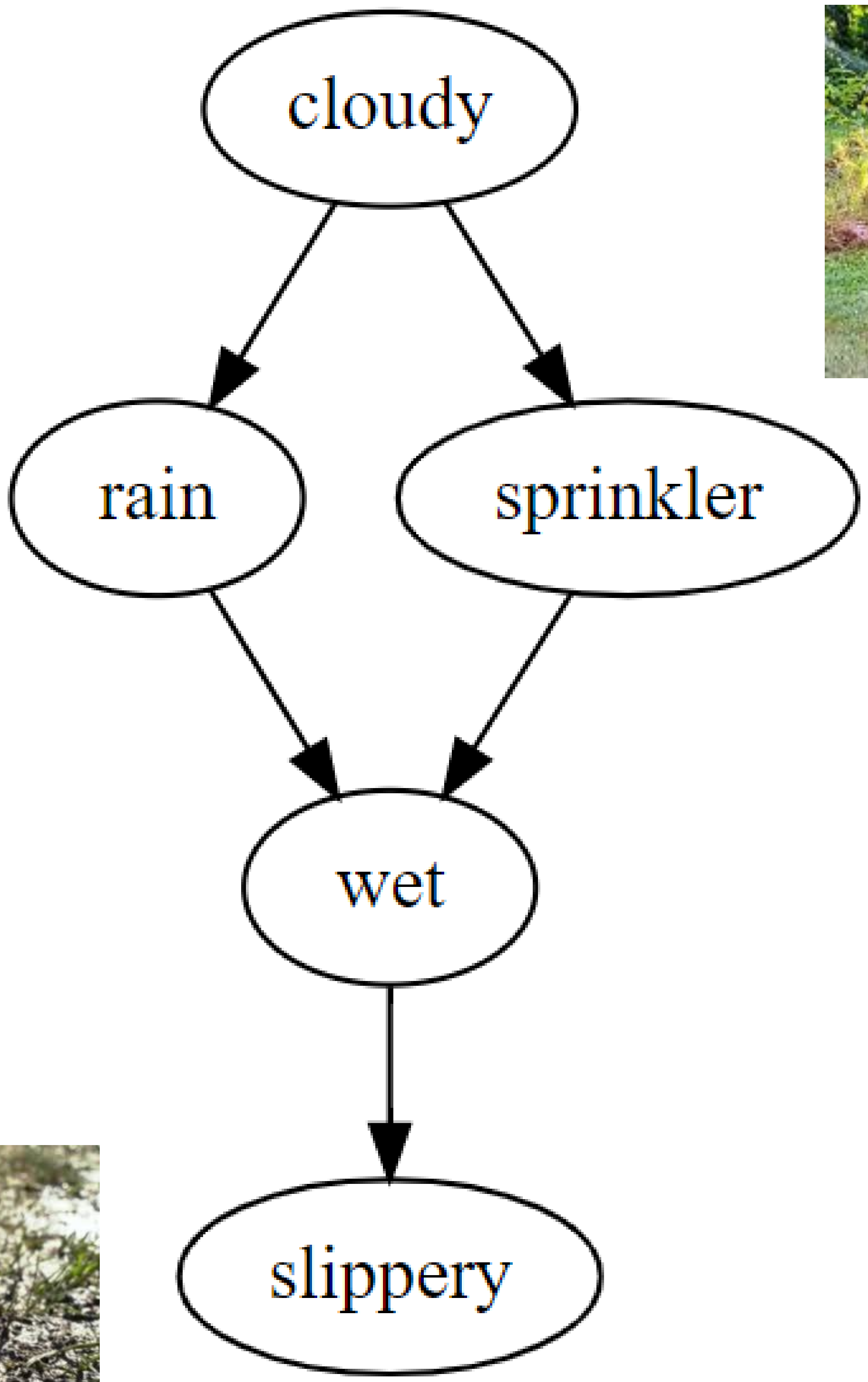
Sometimes, we need to reason about the **structure** of the data generating process itself.

Application of CausalGraphicalModels to the Sprinkler example

Let’s start by introducing the Causal Graphical Models library by applying it to a variant of the Sprinkler example

We consider the following five variables:

- Cloudy: indicates whether it is cloudy
- Rain: indicates whether it is raining
- Sprinkler: indicates whether our sprinkler is on
- Wet: indicates whether the grass is wet
- **Slippery**: indicates whether the ground is slippery



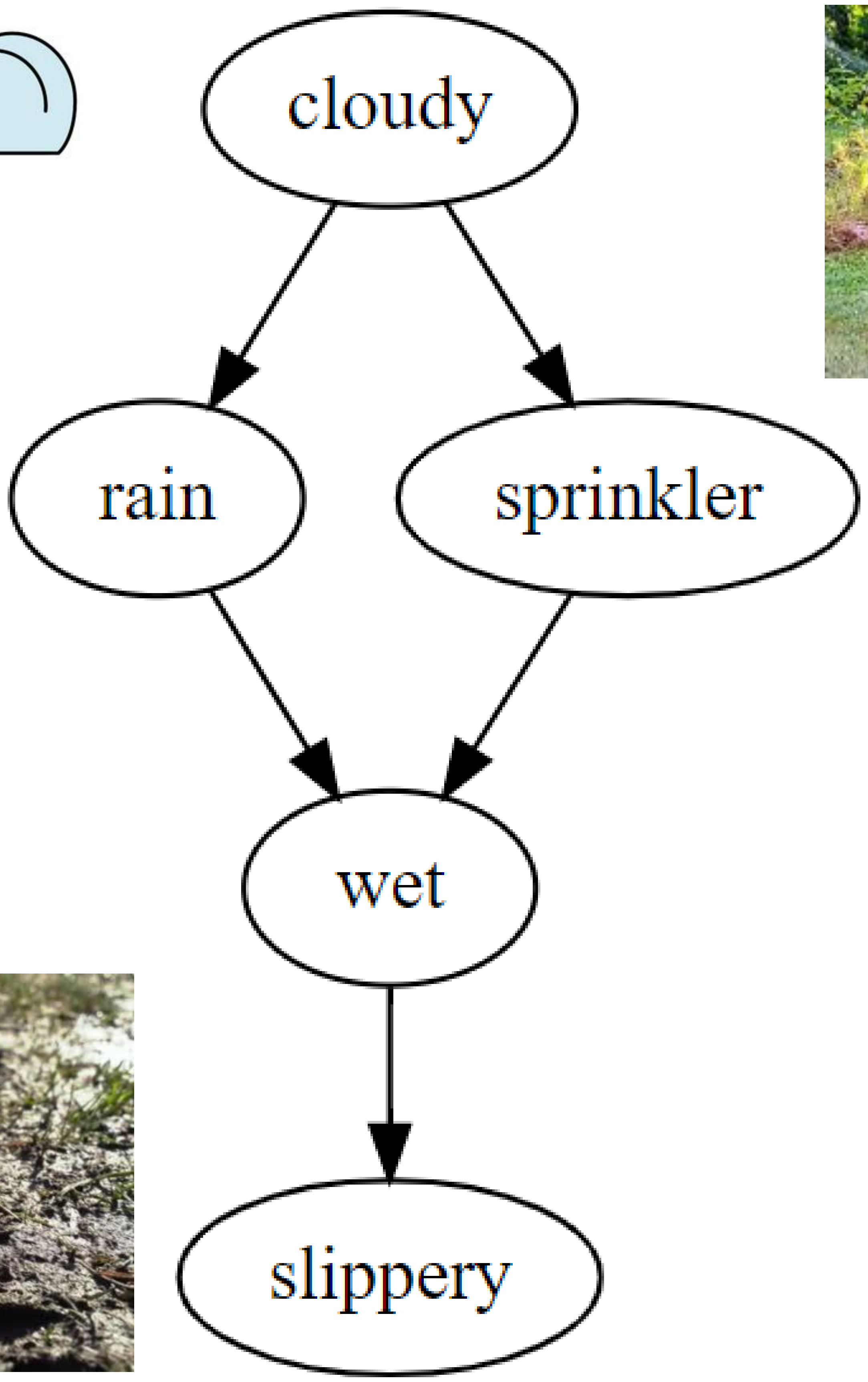
We know that when it rains, the ground will become wet, the making the ground wet doesn't cause it to rain.

Application of CausalGraphicalModels to the Sprinkler example

```
!pip install causalgraphicalmodels
from causalgraphicalmodels import CausalGraphicalModel

sprinkler = CausalGraphicalModel(
    nodes=["cloudy", "rain", "sprinkler", "wet", "slippery"],
    edges=[
        ("cloudy", "rain"),
        ("cloudy", "sprinkler"),
        ("rain", "wet"),
        ("sprinkler", "wet"),
        ("wet", "slippery")
    ]
)

# draw return a graphviz `dot` object, which jupyter can render
sprinkler.draw()
```

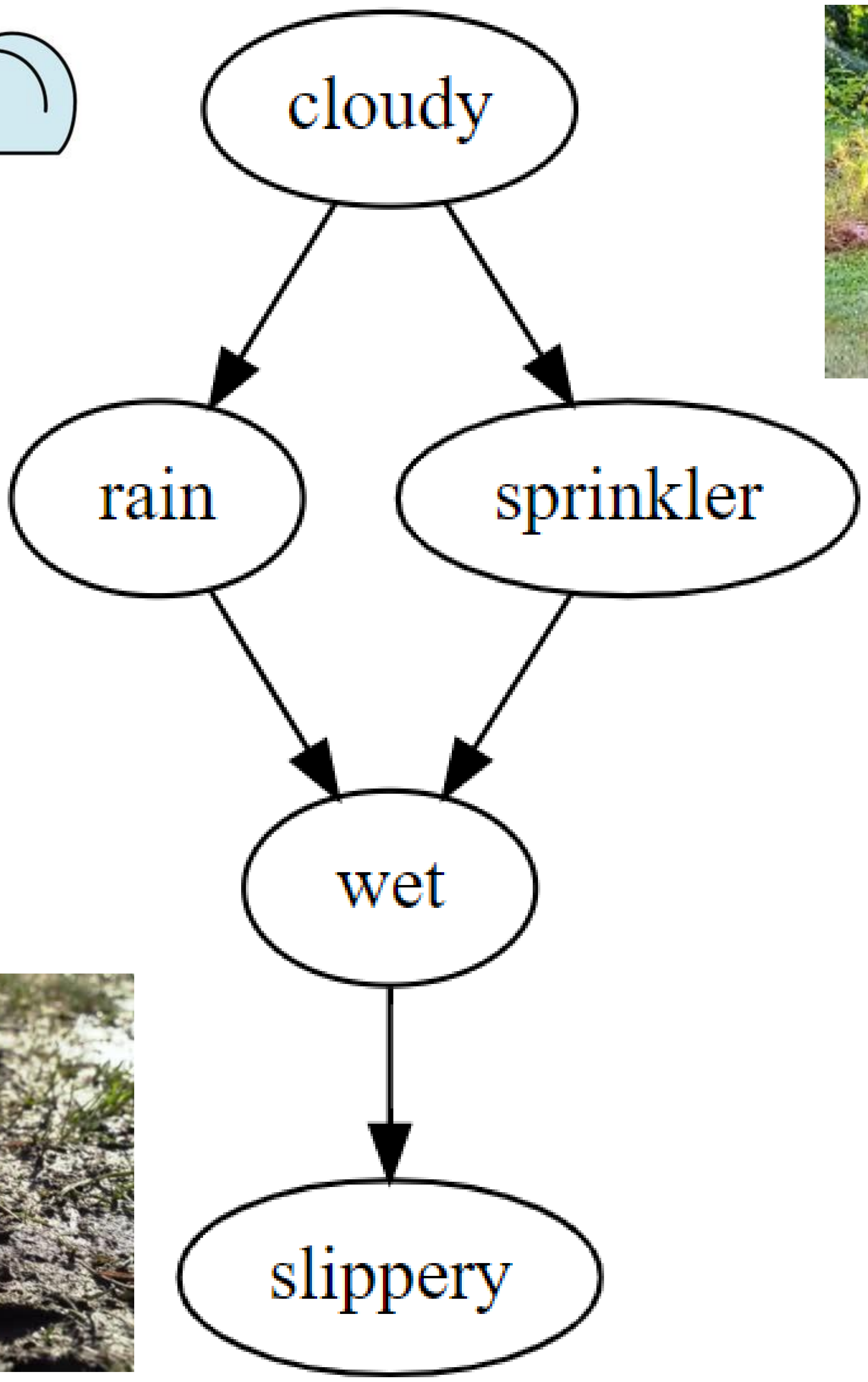


Application of CausalGraphicalModels to the Sprinkler example

```
!pip install causalgraphicalmodels
from causalgraphicalmodels import CausalGraphicalModel

sprinkler = CausalGraphicalModel(
    nodes=["cloudy", "rain", "sprinkler", "wet", "slippery"],
    edges=[
        ("cloudy", "rain"),
        ("cloudy", "sprinkler"),
        ("rain", "wet"),
        ("sprinkler", "wet"),
        ("wet", "slippery")
    ]
)

# draw return a graphviz `dot` object, which jupyter can render
sprinkler.draw()
```



```
# get the distribution implied by the graph
print(sprinkler.get_distribution())
```

```
P(cloudy)P(rain|cloudy)P(sprinkler|cloudy)P(wet|rain,sprinkler)P(slippery|wet)
```

To get the join probability distribution
as we saw in the previous lectures

Application of CausalGraphicalModels to the Sprinkler example

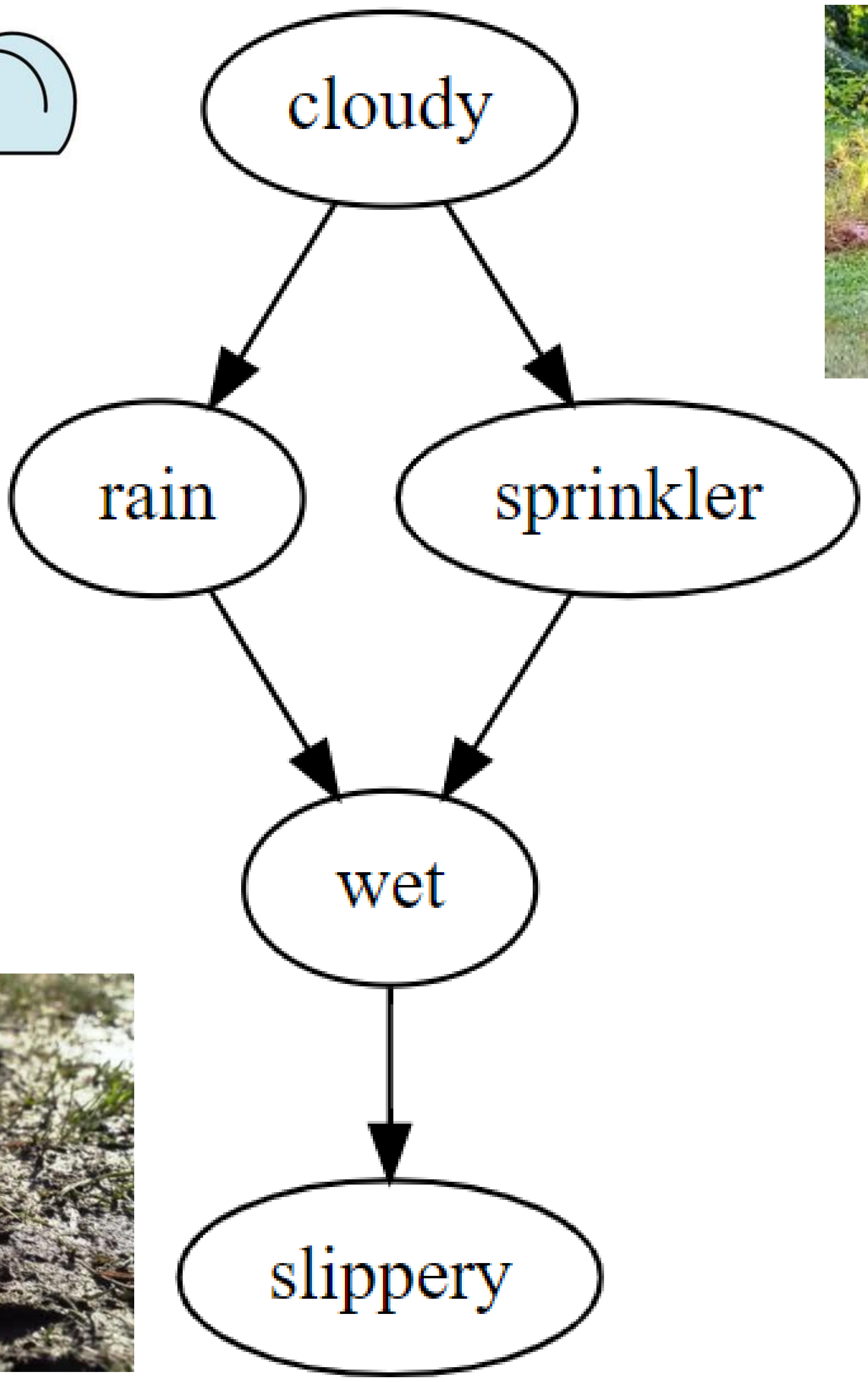
This description of the causal graphical model has been the same as those of general Bayesian Networks.

To endow these structures with a notion of causality, we make some assumptions about what happens when an **intervention** occurs.

For instance, imagine that we have the **power to control the weather**. If we use it to make an intervention on the "rain" node of our sprinkler model, we get the following system.

```
sprinkler_do = sprinkler.do("rain")
```

To apply the intervention



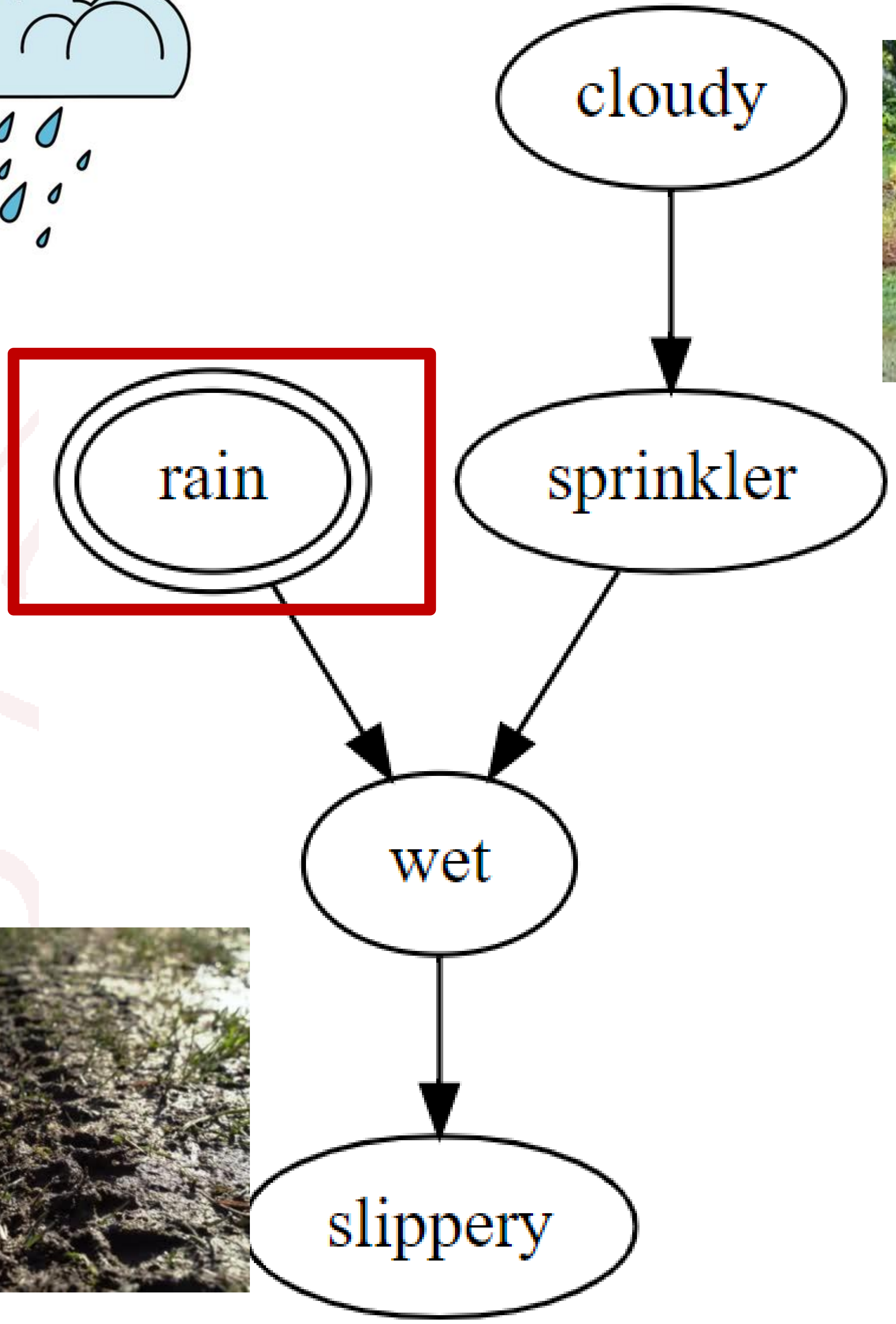
Application of CausalGraphicalModels to the Sprinkler example

For instance, imagine that we have the **power to control the weather**. If we use it to make an intervention on the "rain" node of our sprinkler model, we get the following system.

```
sprinkler_do = sprinkler.do("rain")

sprinkler_do.draw()
```

The visualization has been changed.
A node with a **double outline** indicates a node with an intervention.



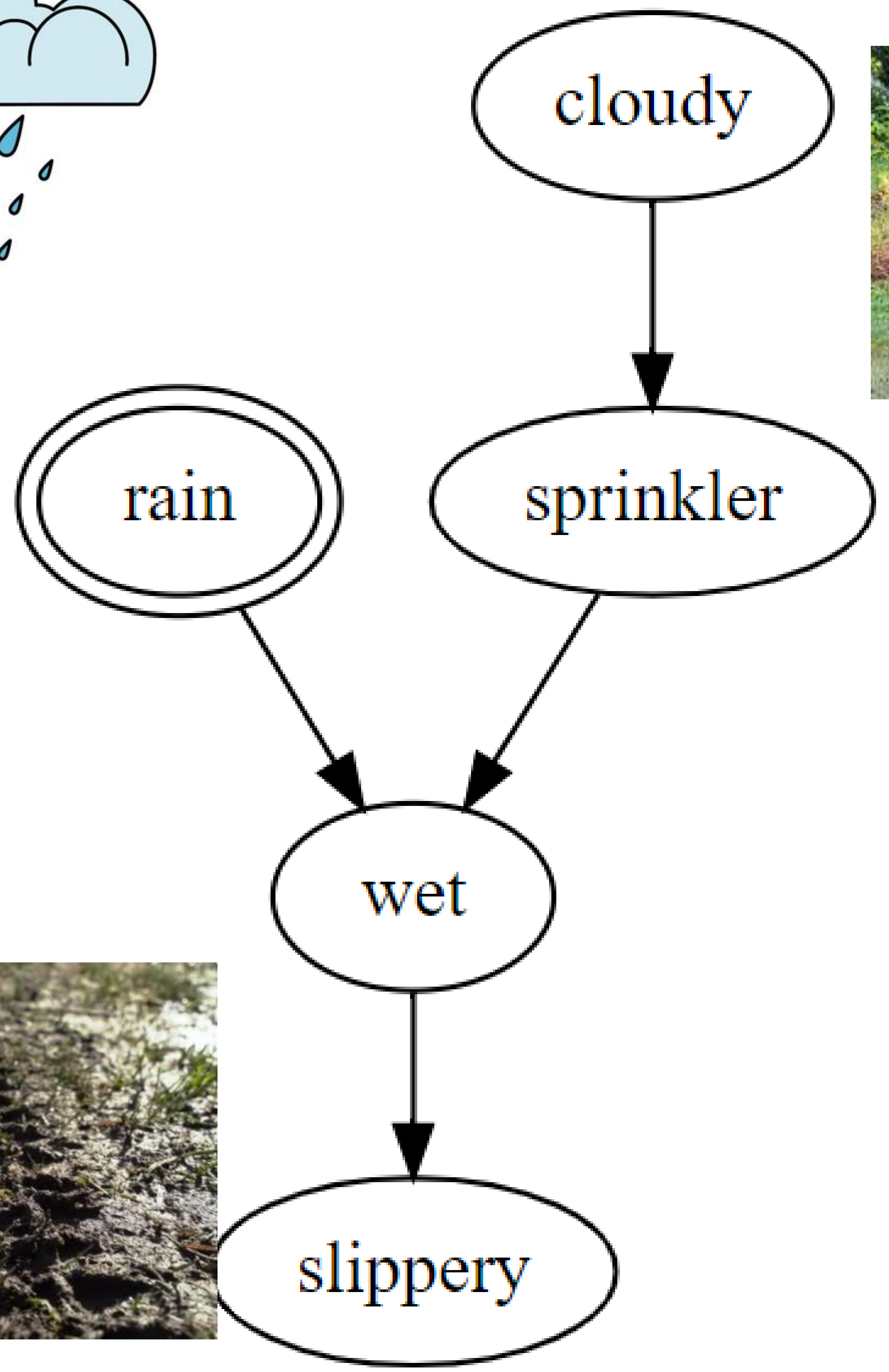
Application of CausalGraphicalModels to the Sprinkler example

For instance, imagine that we have the **power to control the weather**. If we use it to make an intervention on the "rain" node of our sprinkler model, we get the following system.

```
sprinkler_do = sprinkler.do("rain")

sprinkler_do.draw()
```

The visualization has been changed.
A node with a **double outline** indicates a node with an intervention.



```
print(sprinkler_do.get_distribution())
```

```
P(cloudy)P(sprinkler|cloudy)P(wet|do(rain),sprinkler)P(slippery|wet)
```

Obviously, the joint probability distribution is changed too.

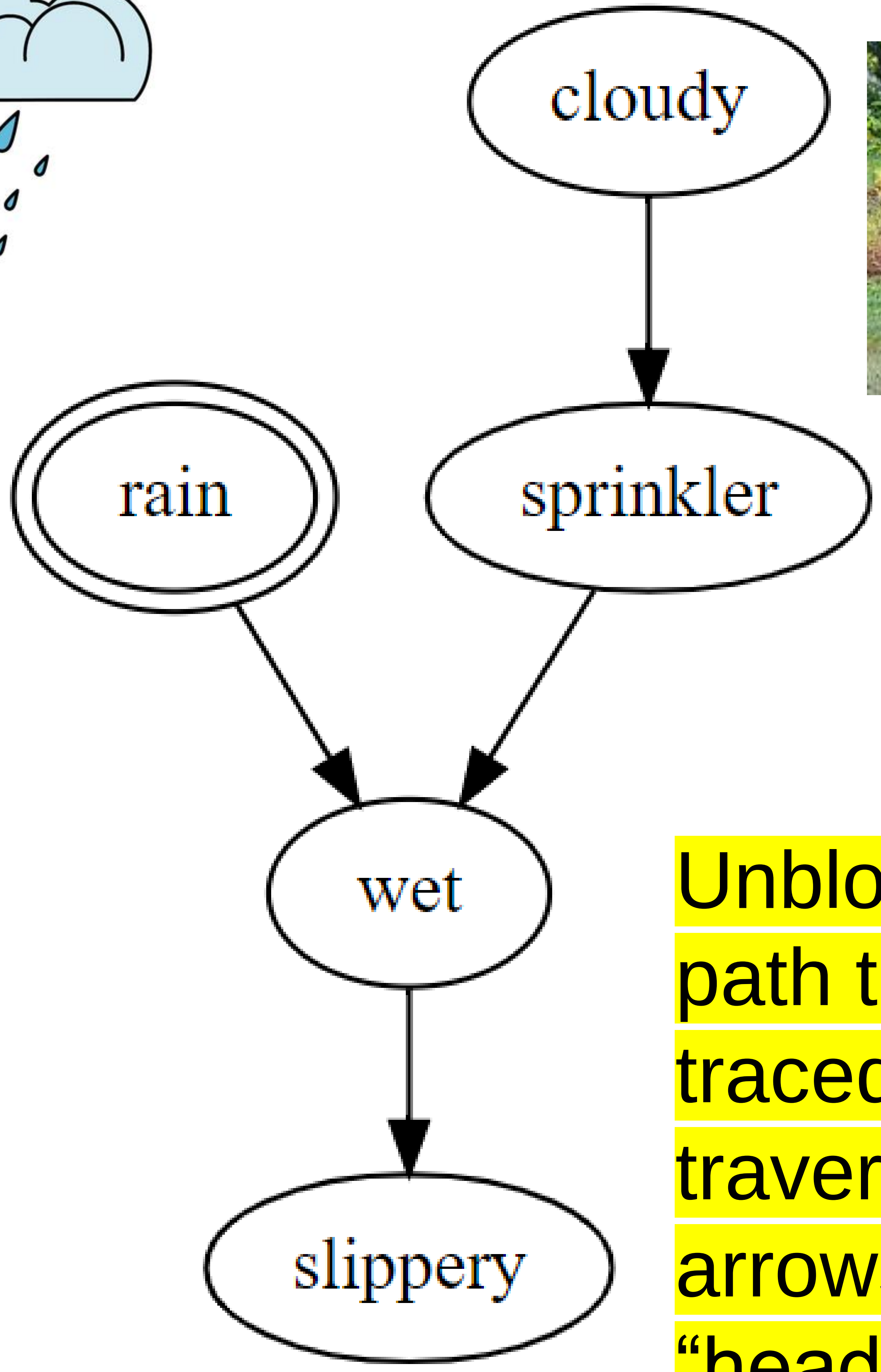
Application of CausalGraphicalModels to the Sprinkler example

Causal graphical models are non-parametric, they cannot tell us what the relationship between two variables are.

They only give us an idea **if there is a relationship** between the two variables through the notion of conditional independence.

It does this using the idea of "paths" between variables: if there are no unblocked paths between two variables, they are independent. It also means that if two causal graphical models share the same paths between two variables, the conditional relationship between these two variables are the same.

For instance, $P(\text{slippery}|\text{wet})$ is the same whether or not we make the intervention on rain, but $P(\text{slippery}|\text{cloudy})$ is not.



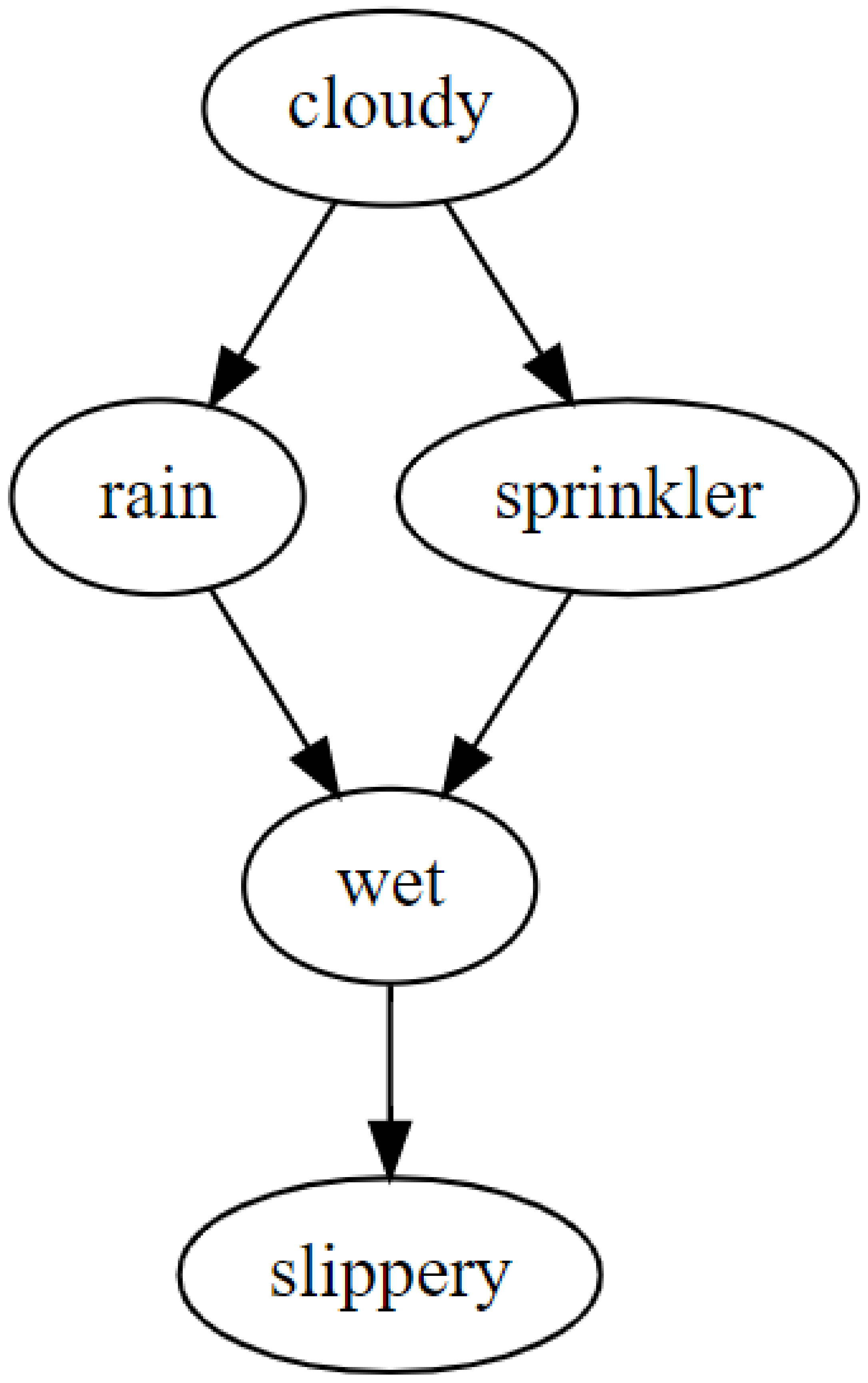
Unblocked path: a path that can be traced without traversing a pair of arrows that collide "head-to-head"

The consequences of interventions

- X can only have some causal inference on Y if there is at least one directed path between X and Y. This is because if there is no directed path, with respect to the intervention graph the parents of X have been removed, so $X \perp\!\!\!\perp Y$ in the intervention graph.

- If there are only directed paths between X and Y, then the causal inference of X and Y is given by the simply conditional distribution $P(Y|X)$. This is because the intervention graph has the same paths between X and Y as the observational distribution.

- If there is a unblocked, but not completely directed path between X and Y, it means that both X and Y share a common ancestor. This common ancestor is what is called as a confounder and this means that if we try to estimate $P(Y|do(X))$ from $P(Y|X)$ of estimates will be biased.



Causal Inference with Causal Graphical Models

Is it possible to make causal inferences from a system we only have observational samples from?

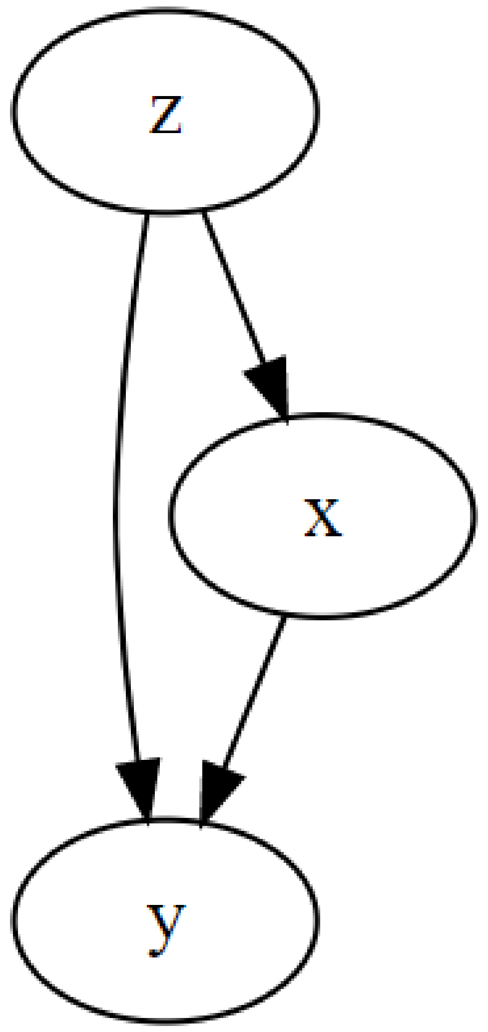
This problem is often called "Identifiability".

Thus, what circumstances can we estimate $P(Y|\, do(X))$ from observational data, given some assumed causal graphical model?

Let’s consider the following simple example:

```
from causalgraphicalmodels.examples import simple_confounded

simple_confounded.draw()
```

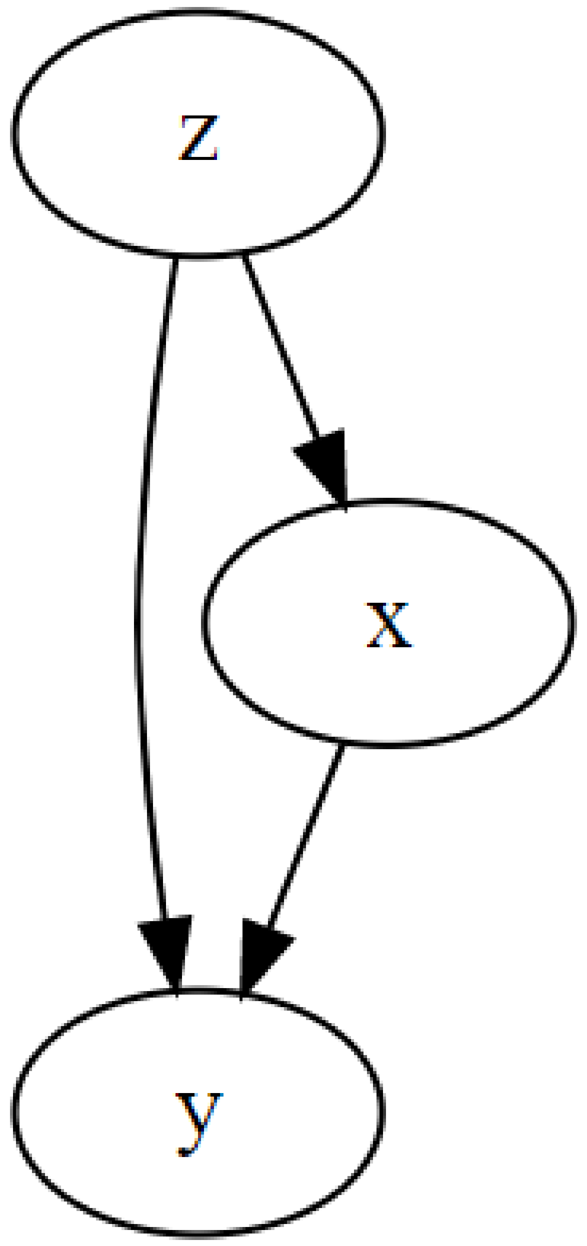


Causal Inference with Causal Graphical Models

Let's consider the following simple example:

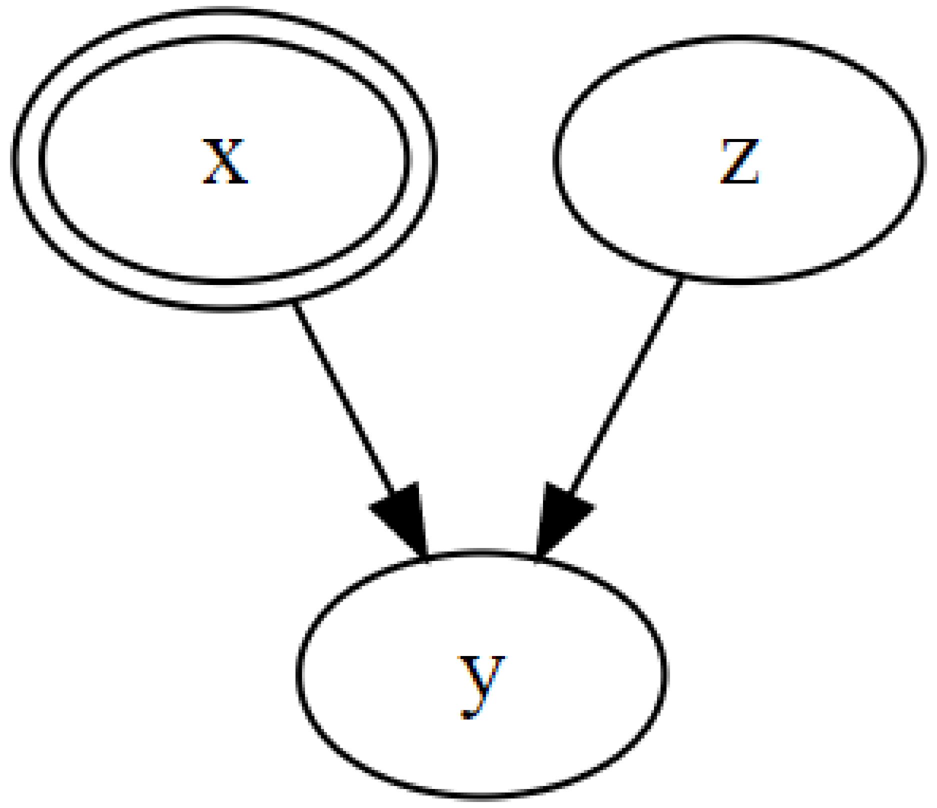
```
from causalgraphicalmodels.examples import simple_confounded

simple_confounded.draw()
```



Under intervention on X, the causal graphical model generating the data is

```
simple_confounded.do("x").draw()
```

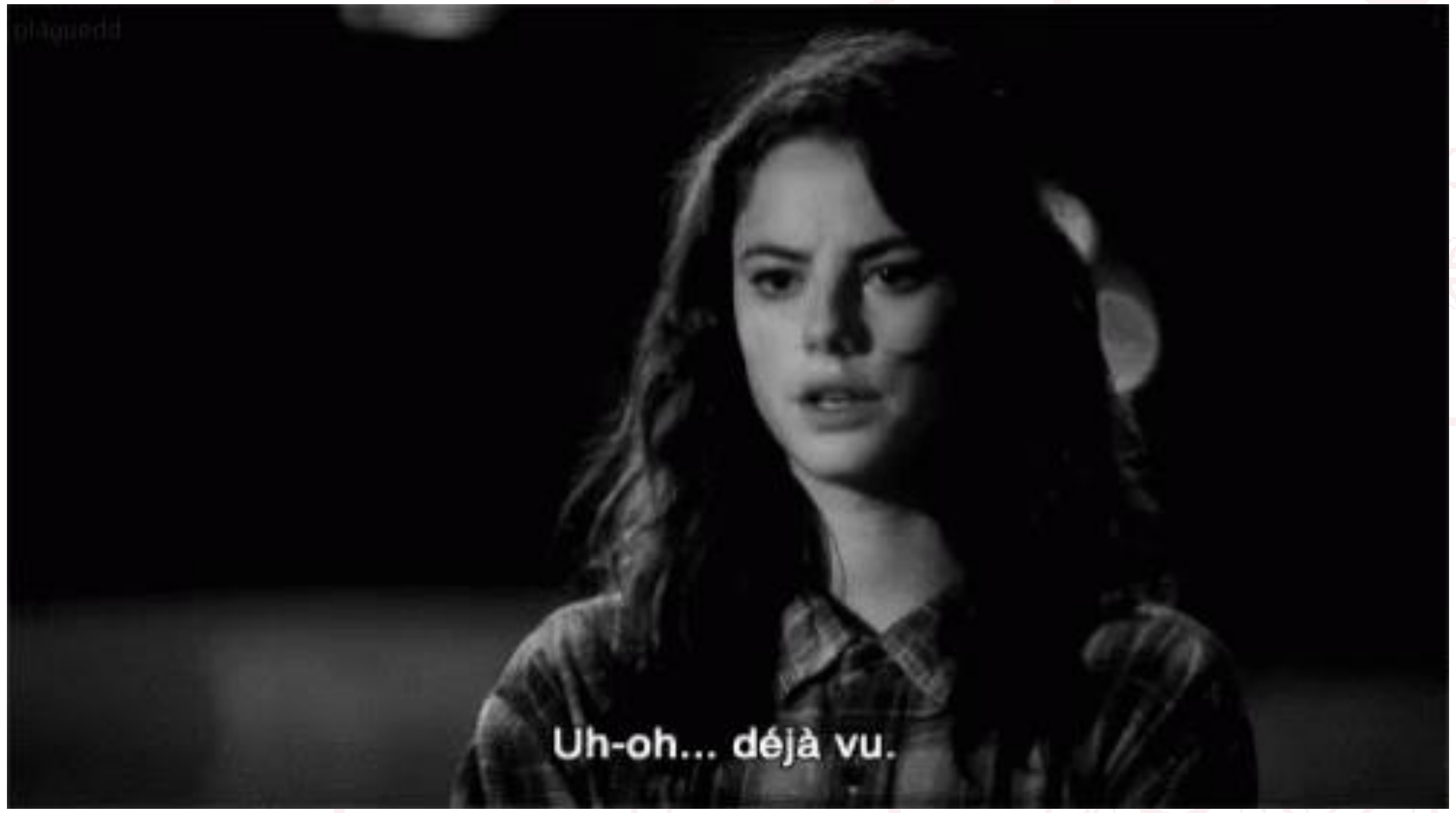


We are going to try and estimate the quantity $P(Y|\, do(X))$

Causal Inference with Causal Graphical Models

We are going to try and estimate the quantity $P(Y|\, do(X))$

We have already seen how to estimate it



Causal Inference with Causal Graphical Models

We are going to try and estimate the quantity $P(Y|\,do(X))$

We have already seen how to estimate it

Adjustment formula

$$\begin{aligned} P(y|do(x)) &= P_m(y|x) && \text{from definition of intervention} \\ &= \sum_z P_m(y|x,z)P_m(z|x) && \text{from Law of Total Probability} \\ &= \sum_z P_m(y|x,z)P_m(z) && \text{from independence of } X \text{ and } Z \\ &= \sum_z P(y|x,z)P(z) && \text{from previous slide's equalities} \end{aligned}$$

- More in general, we can write the **adjustment formula**, or **causal effect rule**:

$$P(y|do(x)) = \sum_{z \in \Lambda} P(y|x,z)P(z)$$

where Λ is the set of parents of X

Slide 33 by prof. Bellotto (Part 1)

Adjustment formula & Backdoor criterion

The adjustment formula states that under certain circumstances, for a set of variables Z, we can estimate the causal influence of X on Y with respect to a causal graphical model using the equation

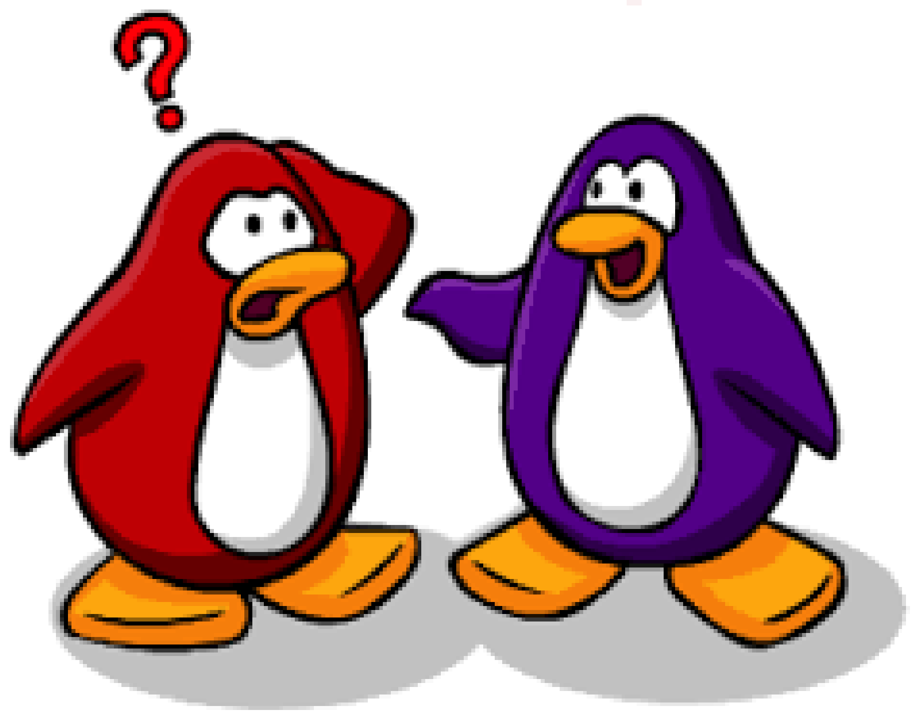
$$P(Y/do(X)) = \sum_Z P(Y|X,Z)\,P(Z)$$

The criterion for Z to exist is sometimes called the **backdoor criterion**.
Graphically it states that:

- Z blocks all backdoor paths between X and Y (all paths with arrows going into X)
- Z does not contain any descendant of X

These criteria are met when Z are the parents of X, but these aren't the only variables which can be used as an adjustment set.

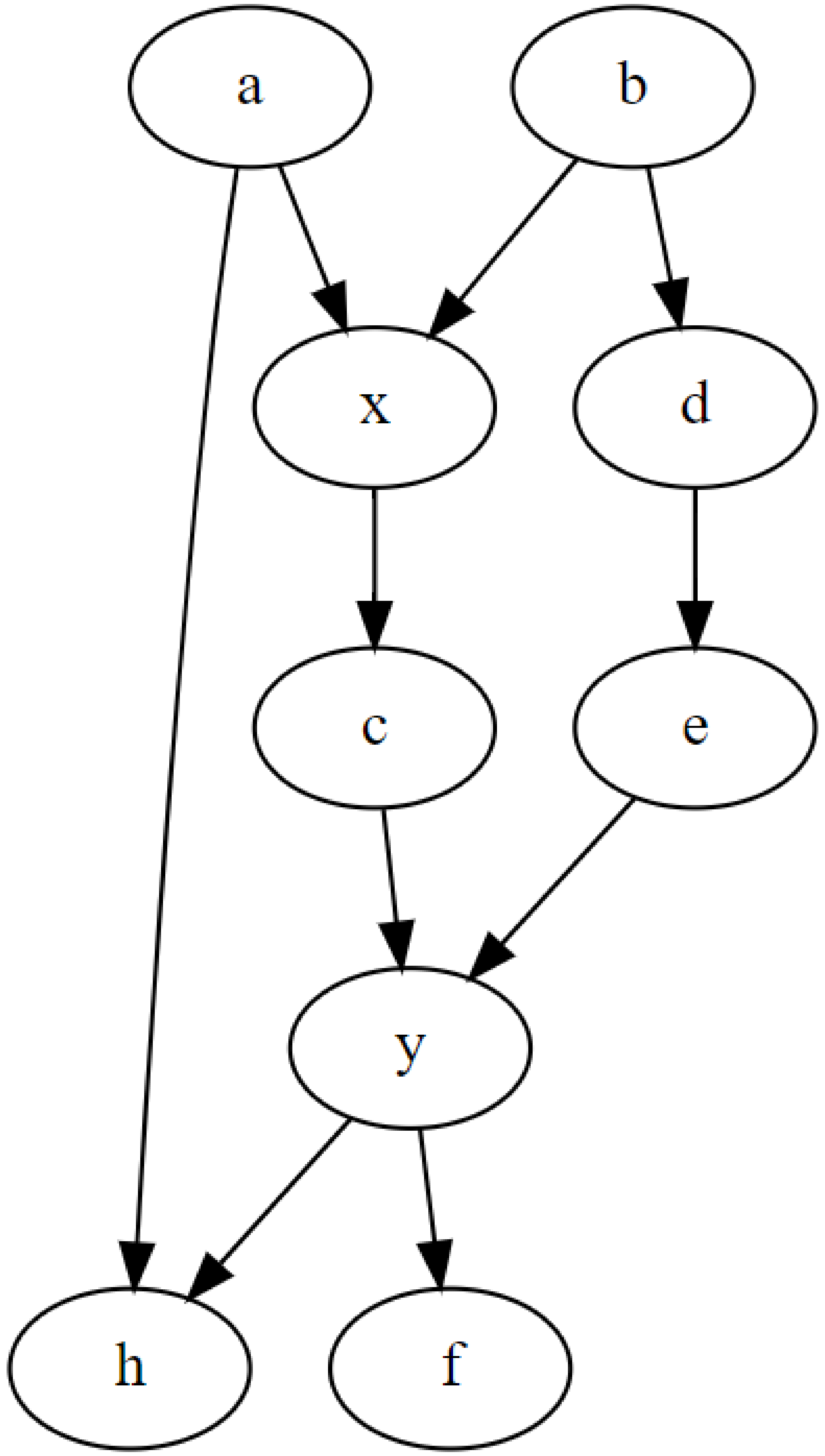
Let’s see an example



Adjustment formula & Backdoor criterion: Example

```python
from causalgraphicalmodels.examples import big_csm

example_cgm = big_csm.cgm
example_cgm.draw()
```

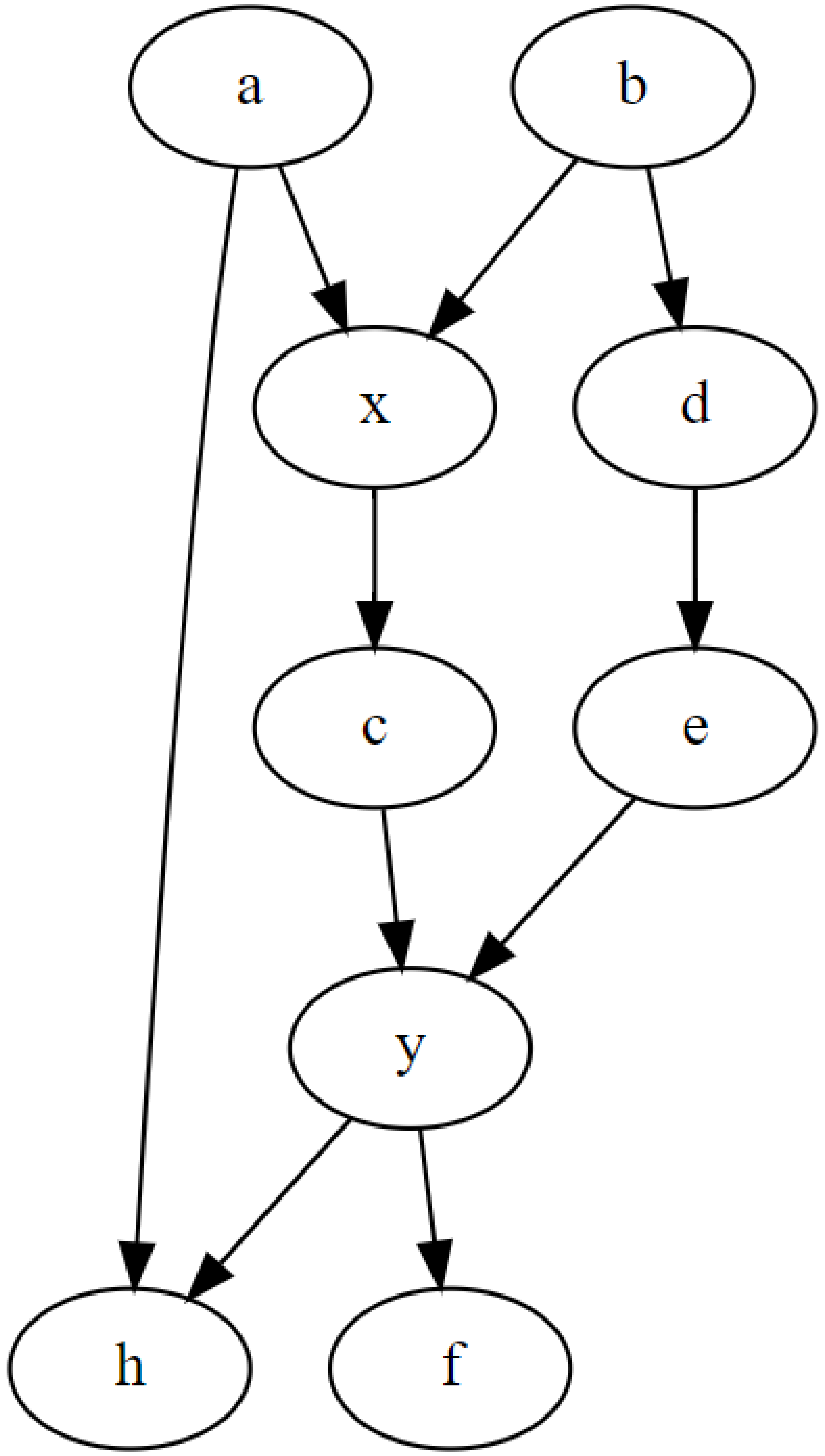


How many backdoor paths are there between X and Y?

Adjustment formula & Backdoor criterion: Example

```
from causalgraphicalmodels.examples import big_csm

example_cgm = big_csm.cgm
example_cgm.draw()
```



How many backdoor paths are there between X and Y ?

```
example_cgm.get_all_backdoor_paths("x", "y")
```

```
[['x', 'a', 'h', 'y'], ['x', 'b', 'd', 'e', 'y']]
```

There are **two** backdoor paths between X and Y

Adjustment formula & Backdoor criterion: Example

```
example_cgm.get_all_backdoor_paths("x", "y")
```

```
[['x', 'a', 'h', 'y'], ['x', 'b', 'd', 'e', 'y']]
```

There are two backdoor paths between X and Y

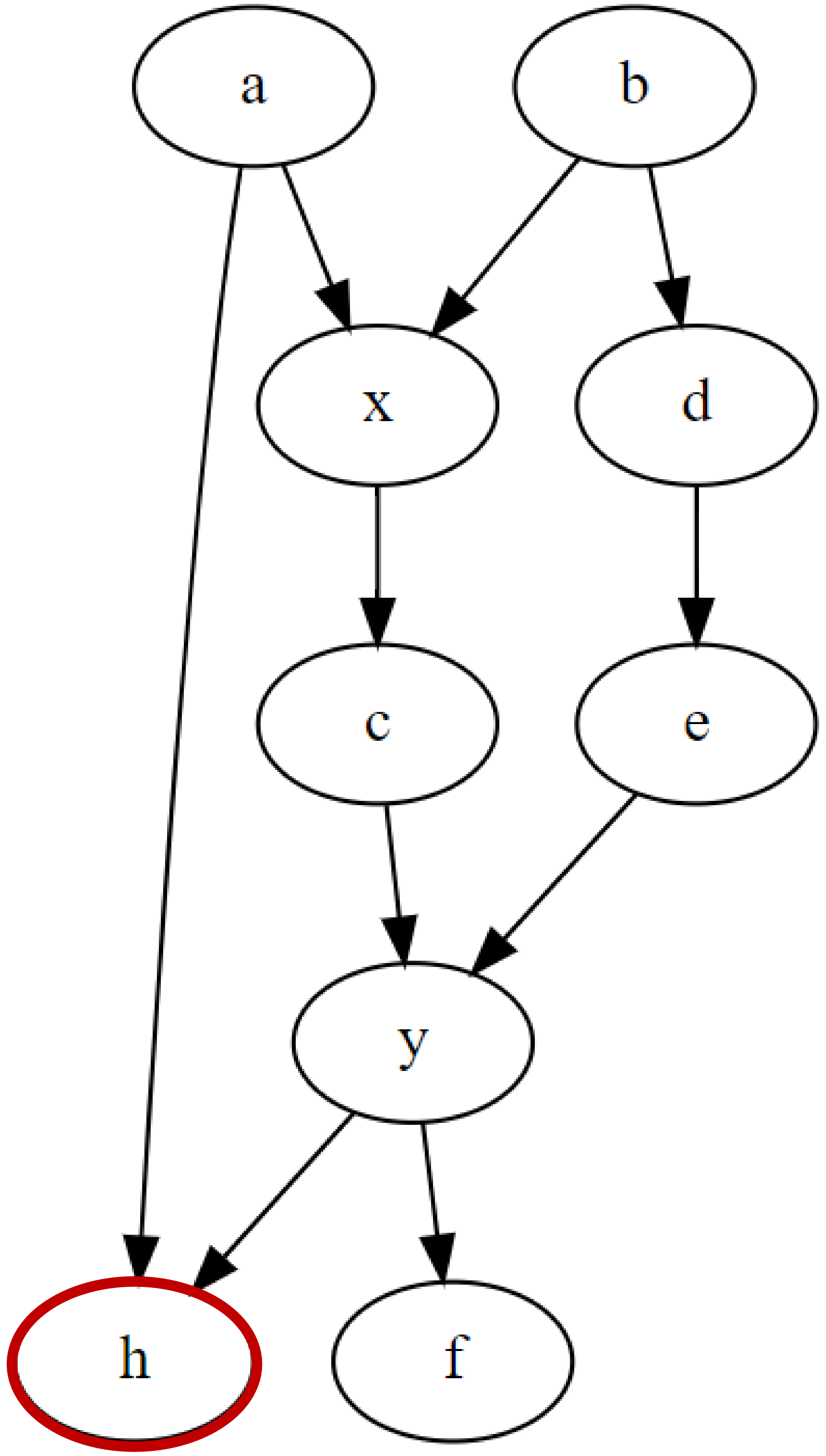
Note: But because h acts as a **collider** in the first path, it is blocked unless conditioned on.

To find a valid adjustment set, we need a set which blocks this path.

Any of the variables B, D, E would work, as well as any combination of the above.

We can also include any other variable in this set, as long as it doesn't create new paths.

Adding H, F or C to the adjustment set would create a new path, making the adjustment set invalid.



Adjustment formula & Backdoor criterion: Example

```
example_cgm.get_all_backdoor_paths("x", "y")
```

```
[['x', 'a', 'h', 'y'], ['x', 'b', 'd', 'e', 'y']]
```

```
example_cgm.is_valid_backdoor_adjustment_set("x", "y", {"a", "h"})
```

```
False
```

```
example_cgm.is_valid_backdoor_adjustment_set("x", "y", {"b", "d", "e"})
```

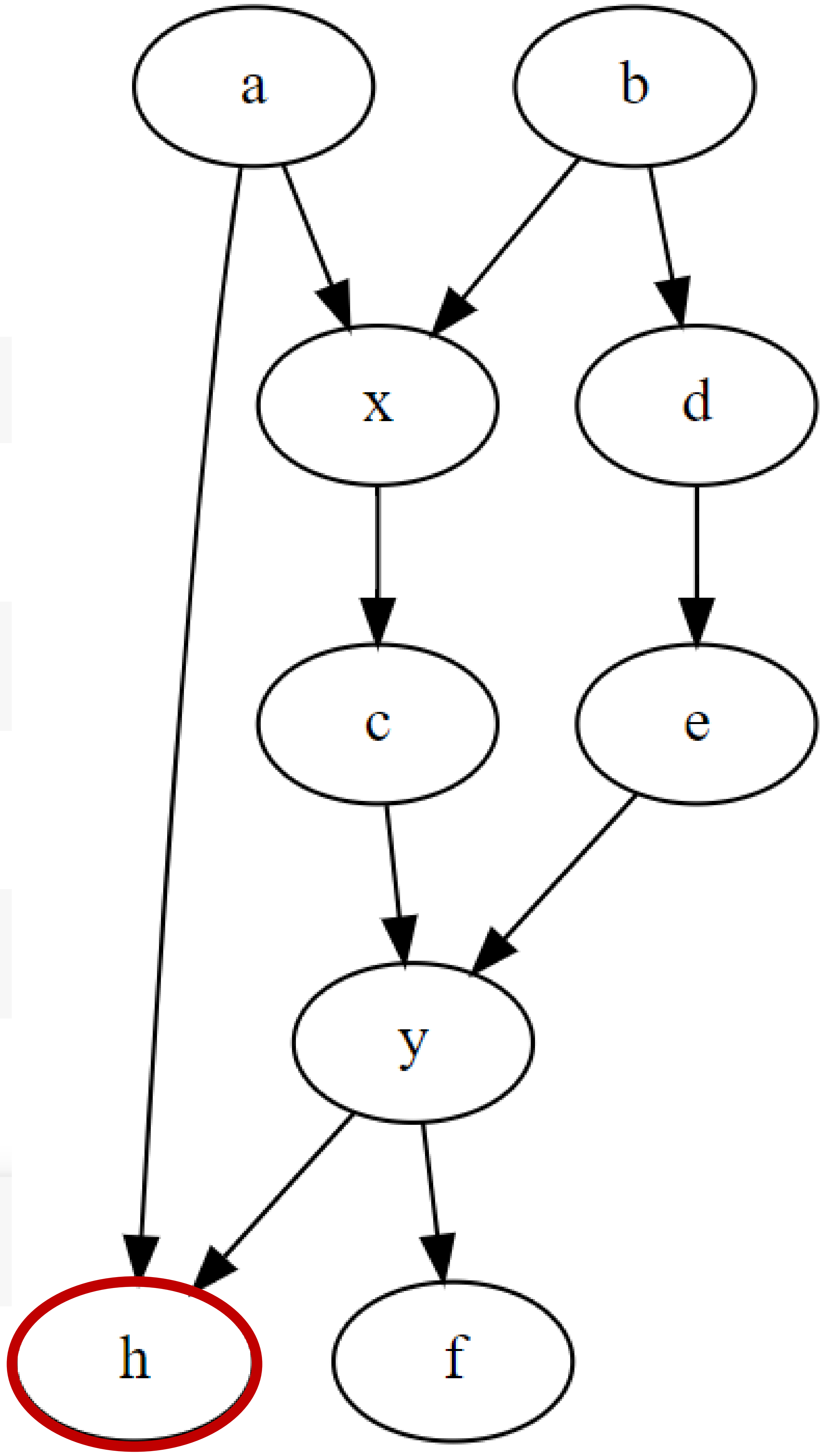
```
True
```

```
example_cgm.is_valid_backdoor_adjustment_set("x", "y", {"b", "d", "e", "h"})
```

```
False
```

```
example_cgm.is_valid_backdoor_adjustment_set("x", "y", {"b", "d", "e", "f"})
```

```
False
```

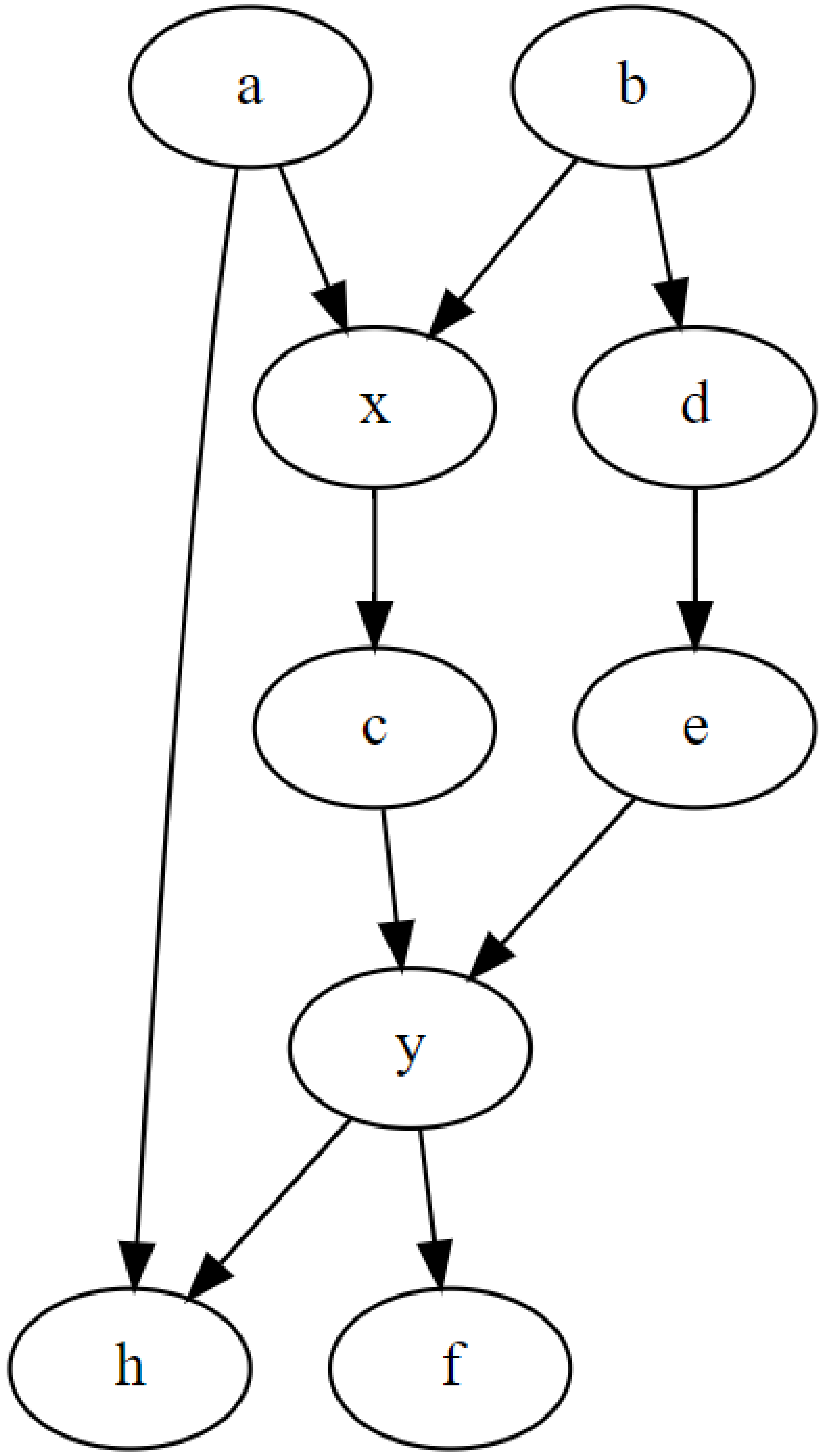


Adjustment formula & Backdoor criterion: Example

We can compute all valid adjustment sets using the following:

```
example_cgm.get_all_backdoor_adjustment_sets("x", "y")
```

```
frozenset({frozenset({'b'}),
           frozenset({'e'}),
           frozenset({'b', 'e'}),
           frozenset({'a', 'd'}),
           frozenset({'a', 'd', 'e'}),
           frozenset({'a', 'b'}),
           frozenset({'b', 'd'}),
           frozenset({'a', 'b', 'd'}),
           frozenset({'a', 'b', 'e'}),
           frozenset({'d'}),
           frozenset({'d', 'e'}),
           frozenset({'b', 'd', 'e'}),
           frozenset({'a', 'e'}),
           frozenset({'a', 'b', 'd', 'e'})})
```

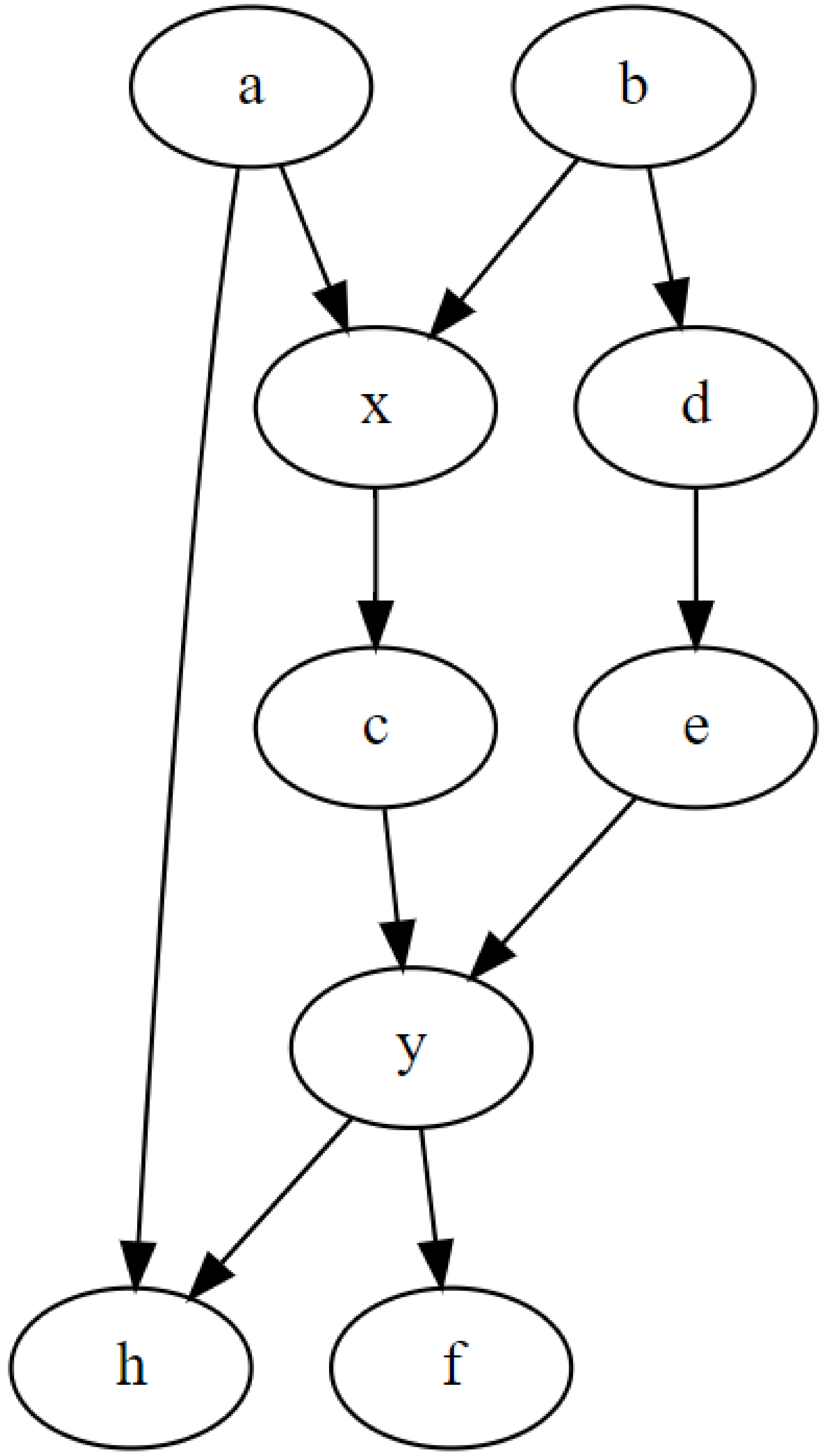


Adjustment formula & Backdoor criterion: Example

We can compute all valid adjustment sets using the following:

```
example_cgm.get_all_backdoor_adjustment_sets("x", "y")
```

```
frozenset({frozenset({'b'}),
           frozenset({'e'}),
           frozenset({'b', 'e'}),
           frozenset({'a', 'd'}),
           frozenset({'a', 'd', 'e'}),
           frozenset({'a', 'b'}),
           frozenset({'b', 'd'}),
           frozenset({'a', 'b', 'd'}),
           frozenset({'a', 'b', 'e'}),
           frozenset({'d'}),
           frozenset({'d', 'e'}),
           frozenset({'b', 'd', 'e'}),
           frozenset({'a', 'e'}),
           frozenset({'a', 'b', 'd', 'e'})})
```



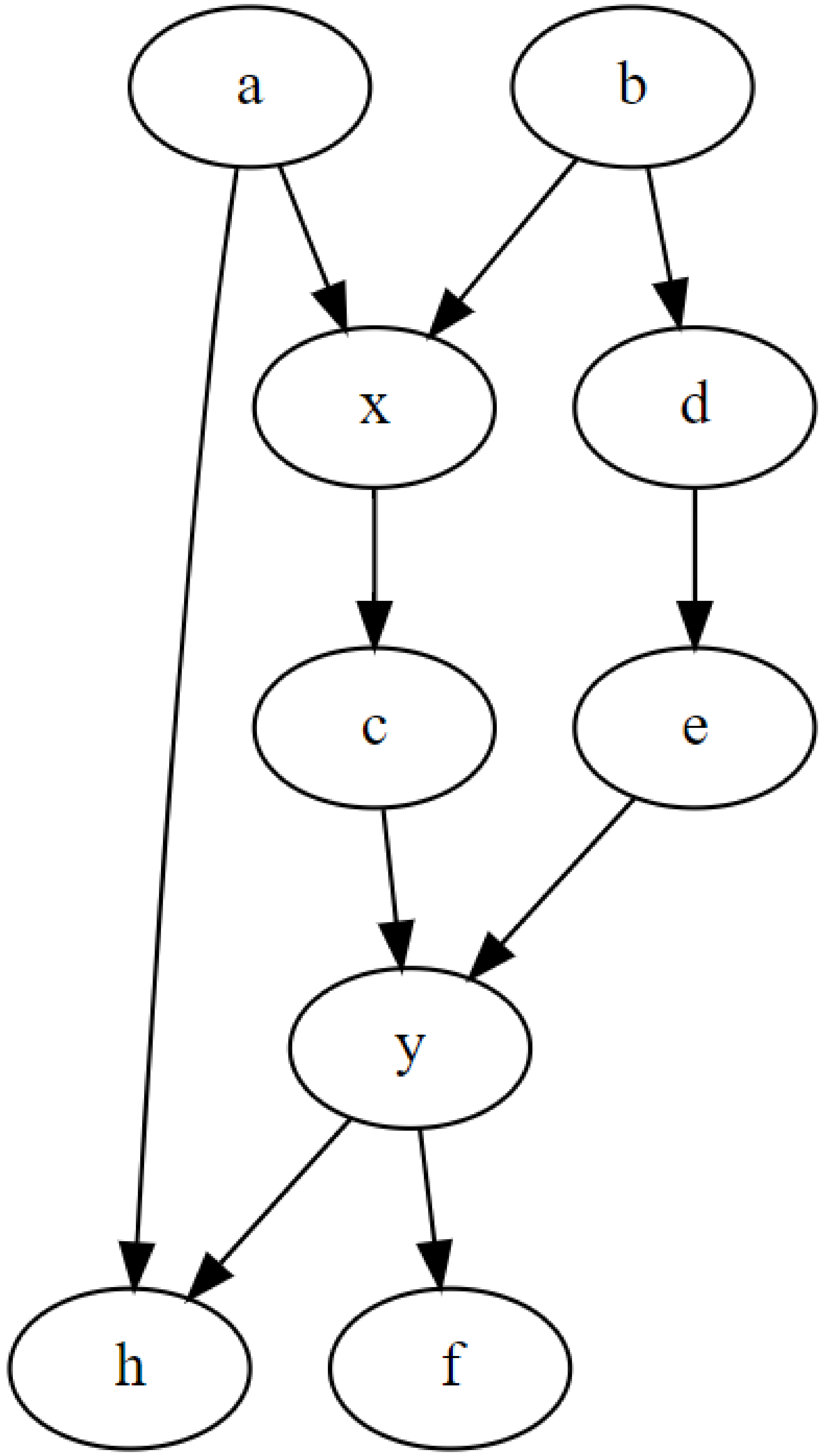
Note: blocking all backdoor paths accounts for any bias introduced by confounding variables, and the requirement that no descendants are conditioned on prevents any new paths from being created.

Adjustment formula & Backdoor criterion: Example

We can compute all valid adjustment sets using the following:

```
example_cgm.get_all_backdoor_adjustment_sets("x", "y")
```

```
frozenset({frozenset({'b'}),
           frozenset({'e'}),
           frozenset({'b', 'e'}),
           frozenset({'a', 'd'}),
           frozenset({'a', 'd', 'e'}),
           frozenset({'a', 'b'}),
           frozenset({'b', 'd'}),
           frozenset({'a', 'b', 'd'}),
           frozenset({'a', 'b', 'e'}),
           frozenset({'d'}),
           frozenset({'d', 'e'}),
           frozenset({'b', 'd', 'e'}),
           frozenset({'a', 'e'}),
           frozenset({'a', 'b', 'd', 'e'})})
```



Note: When all variables in a causal graphical model are observed, is always a set which can be used for adjustment. If not all variables are observed, there can be causal statements which cannot be estimated from the observed data.

Unobserved confounders: Smoking example

We have seen how causal graphical models can be used as a way to make statements about *causal inference* (how data would be generated if there is an intervention on the system),

whereas potential outcomes describe *counterfactual inference* (what would have happened to a system which had already been observed, if a different treatment had been applied),

but it is possible to use CGMs/SCMs to reason about unobserved confounders.

Unobserved confounders: Smoking example

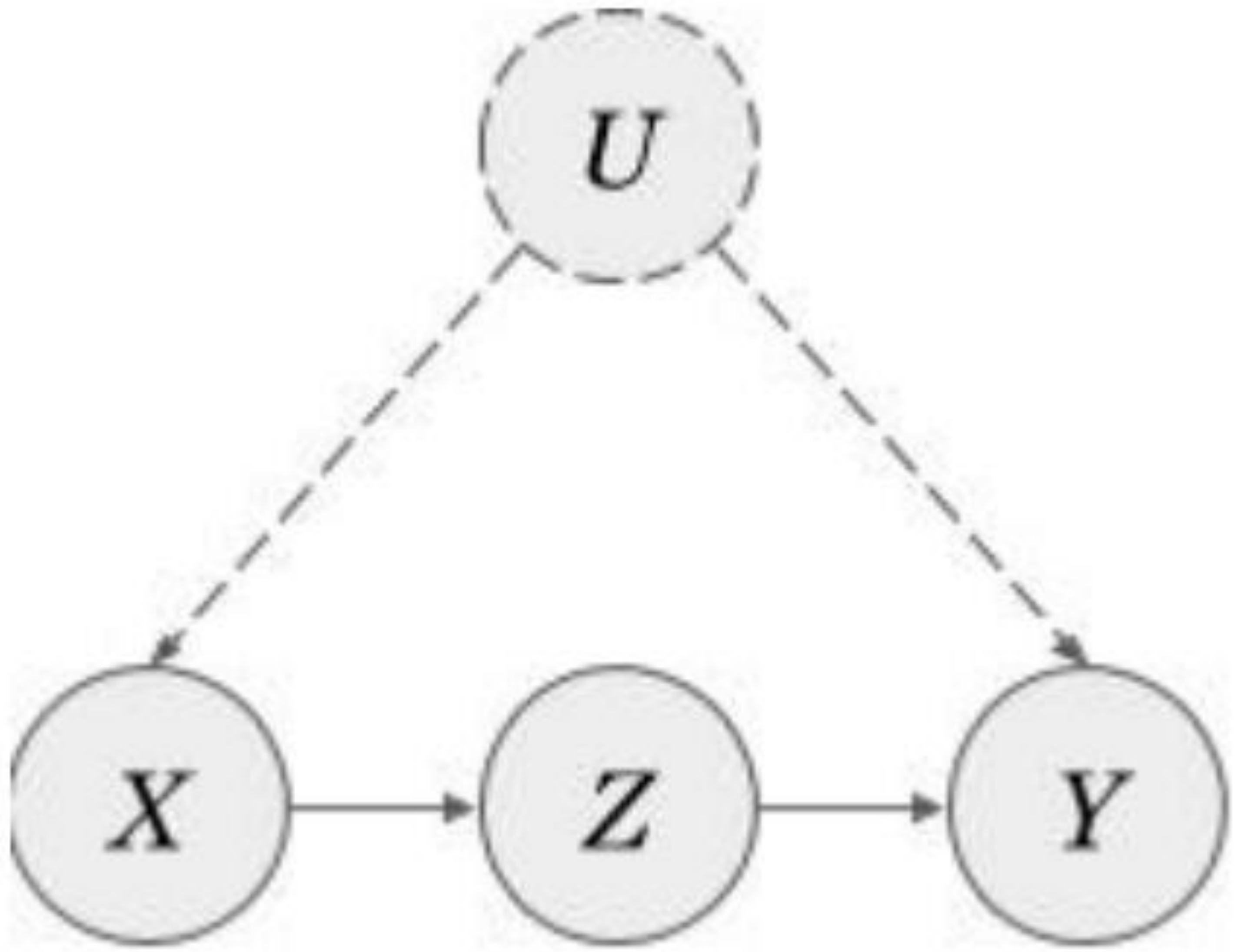
We have seen how causal graphical models can be used as a way to make statements about *causal inference* (how data would be generated if there is an intervention on the system),

whereas potential outcomes describe *counterfactual inference* (what would have happened to a system which had already been observed, if a different treatment had been applied),

but it is possible to use CGMs/SCMs to reason about unobserved confounders.

Let's consider the following smoking example:

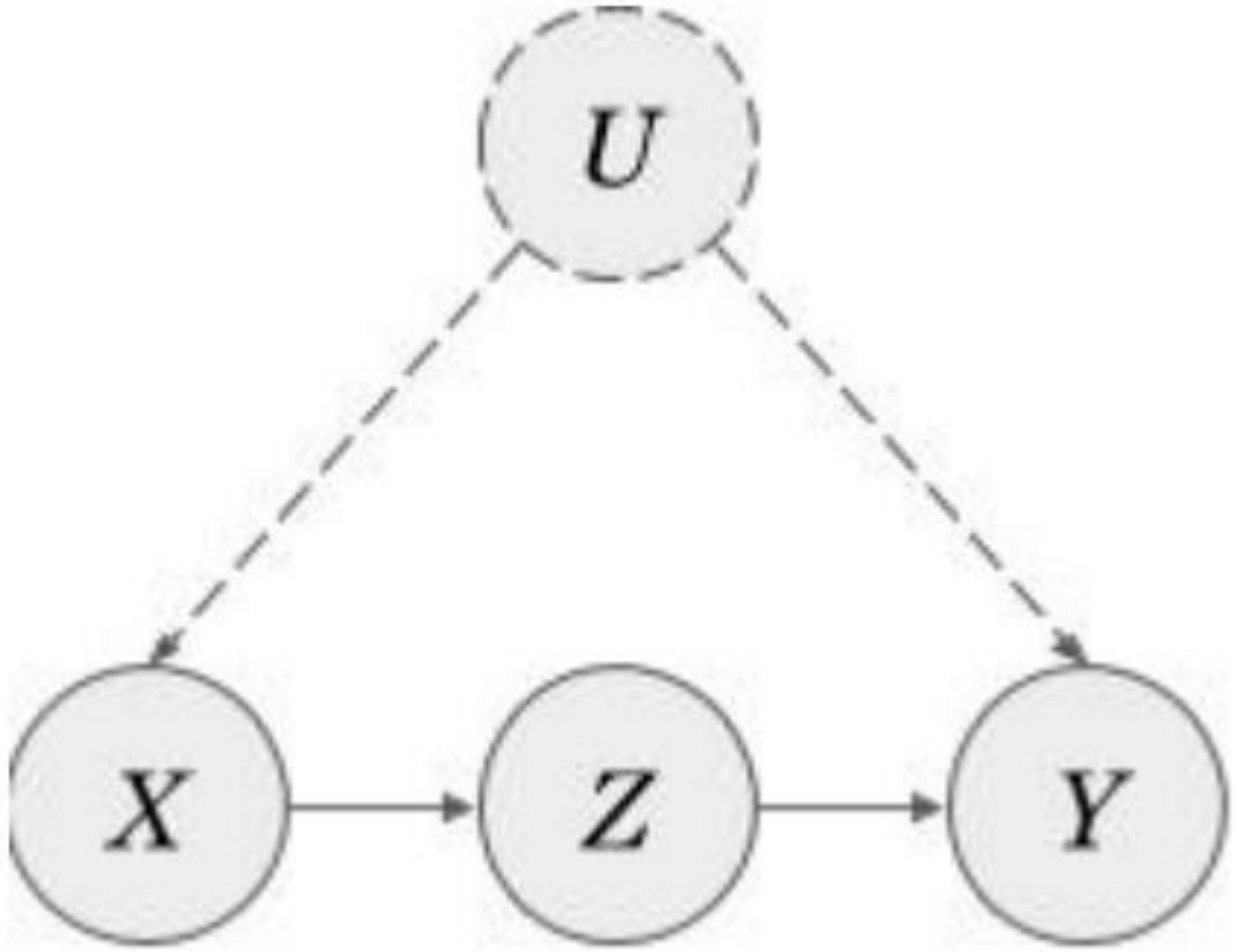
- U (smoking gene) → unobserved confounders
- Z (tar deposit) → observed variables,
- X (smoking) causes Z (tar deposit)
- Z (tar deposit) outcomes Y (lung cancer)



Unobserved confounders: Smoking example

Let's consider the following smoking example:

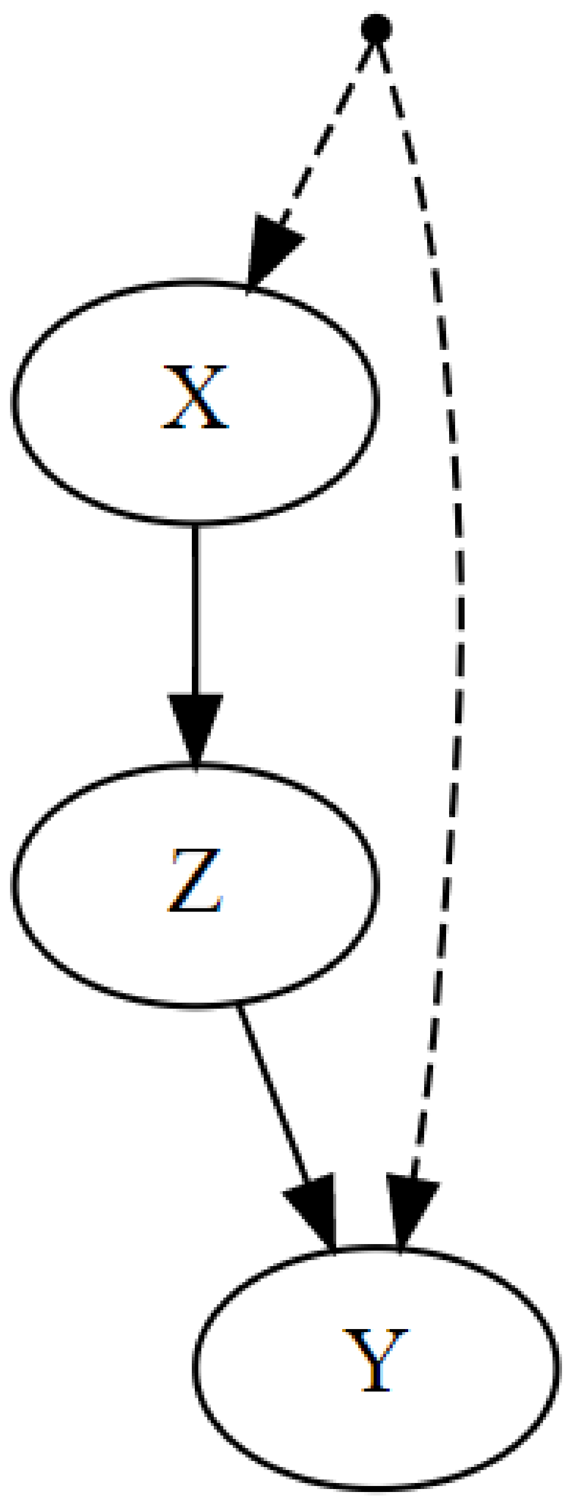
- U (smoking gene) → unobserved confounders
- Z (tar deposit) → observed variables,
- X (smoking) causes Z (tar deposit)
- Z (tar deposit) outcomes Y (lung cancer)



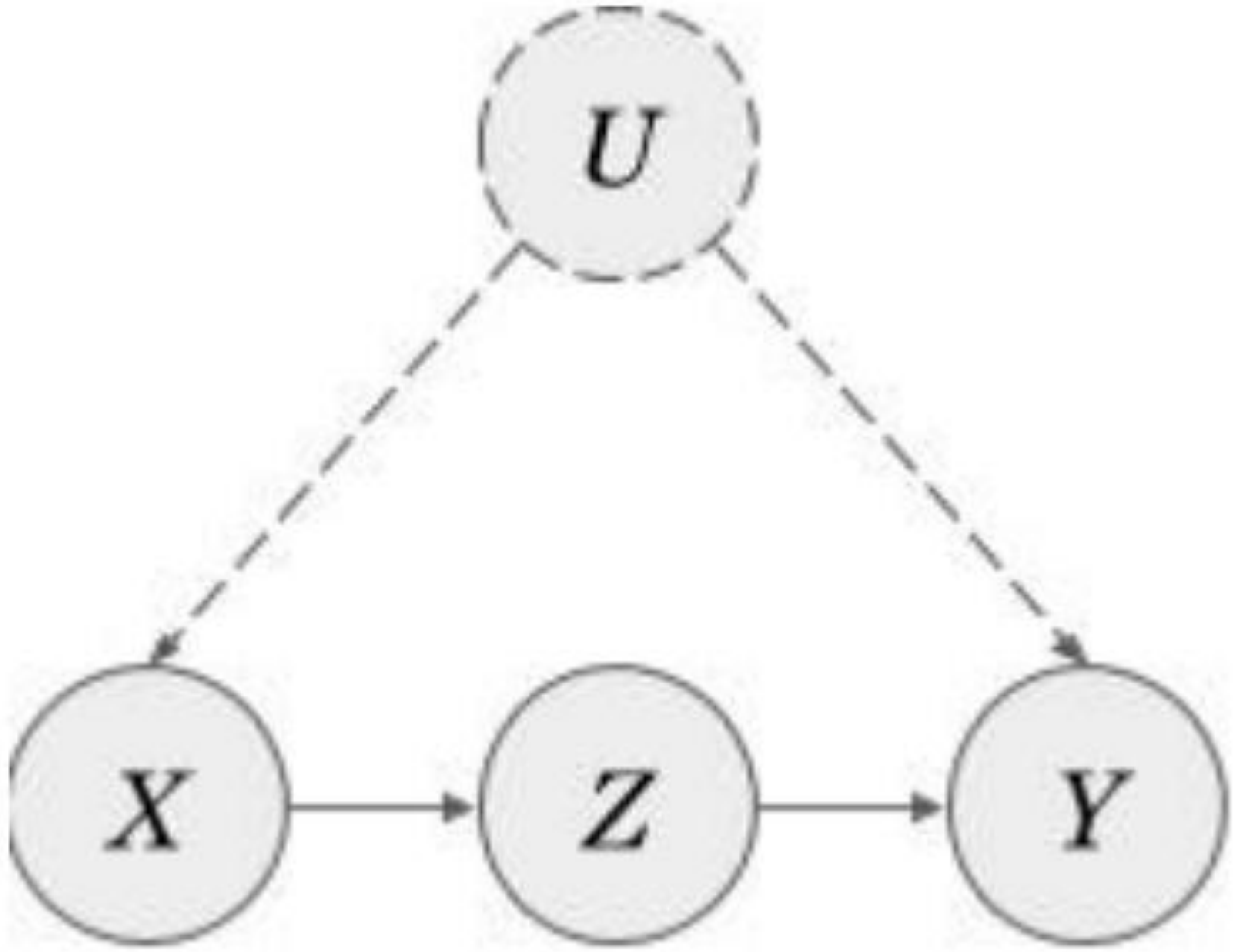

```
bdnet3 = CausalGraphicalModel(
    nodes = ['X', 'Y', 'Z'],
    edges = [
        ('X','Z'),
        ('Z', 'Y')
    ],
    latent_edges = [
    ('X', 'Y')
    ]

)

bdnet3.draw()
```



Unobserved confounders: Smoking example



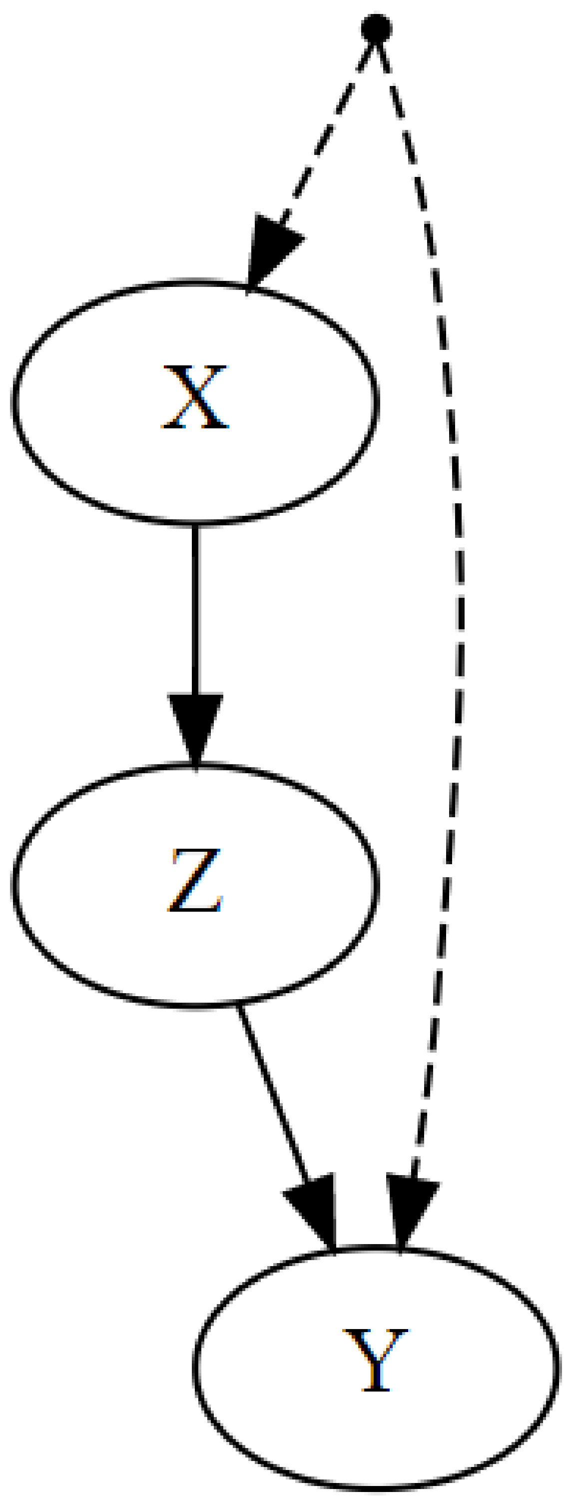
Let's consider the following smoking example:

- U (smoking gene) → unobserved confounders
- Z (tar deposit) → observed variables,
- X (smoking) causes Z (tar deposit)
- Z (tar deposit) outcomes Y (lung cancer)

```
bdnet3 = CausalGraphicalModel(
    nodes = ['X', 'Y', 'Z'],
    edges = [
        ('X','Z'),
        ('Z', 'Y')
    ],
    latent_edges = [
    ('X', 'Y')
    ]

)

bdnet3.draw()
```



where the variable U, since unobserved, is substituted by a **generic confounder link** (**dashed arrows**) between X and Y

Front-door adjustment

To calculate the causal effect of smoking (X) on lung cancer (Y) by observing the presence of tar deposit (Z), even without knowing if there are other confounders (i.e. smoking gene U)

Front-door adjustment

- A set **S** of variables satisfies the **front-door criterion** if 1) it blocks every directed path between X and Y, 2) there are no backdoor paths between X and **S**, and 3) all the backdoor paths from **S** to Y are blocked by X
- If **S** satisfies the front-door criterion, then the following formula can be used to compute the causal effect of X on Y (**front-door adjustment**)

$$P(y|do(x)) = \sum_{s} P(s|x) \sum_{x'} P(y|x', s) P(x')$$

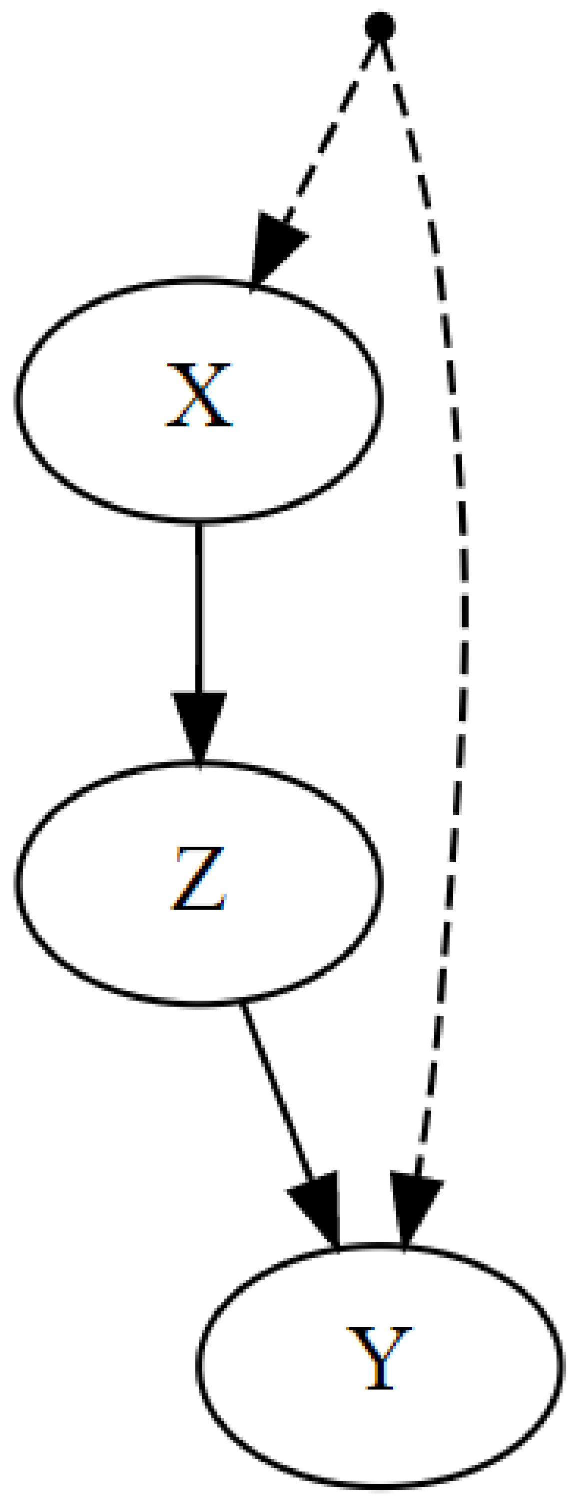
where s are all the possible values of the variables in **S**

Slide 22 by prof. Bellotto (Part 2)

Unobserved confounders: Smoking example

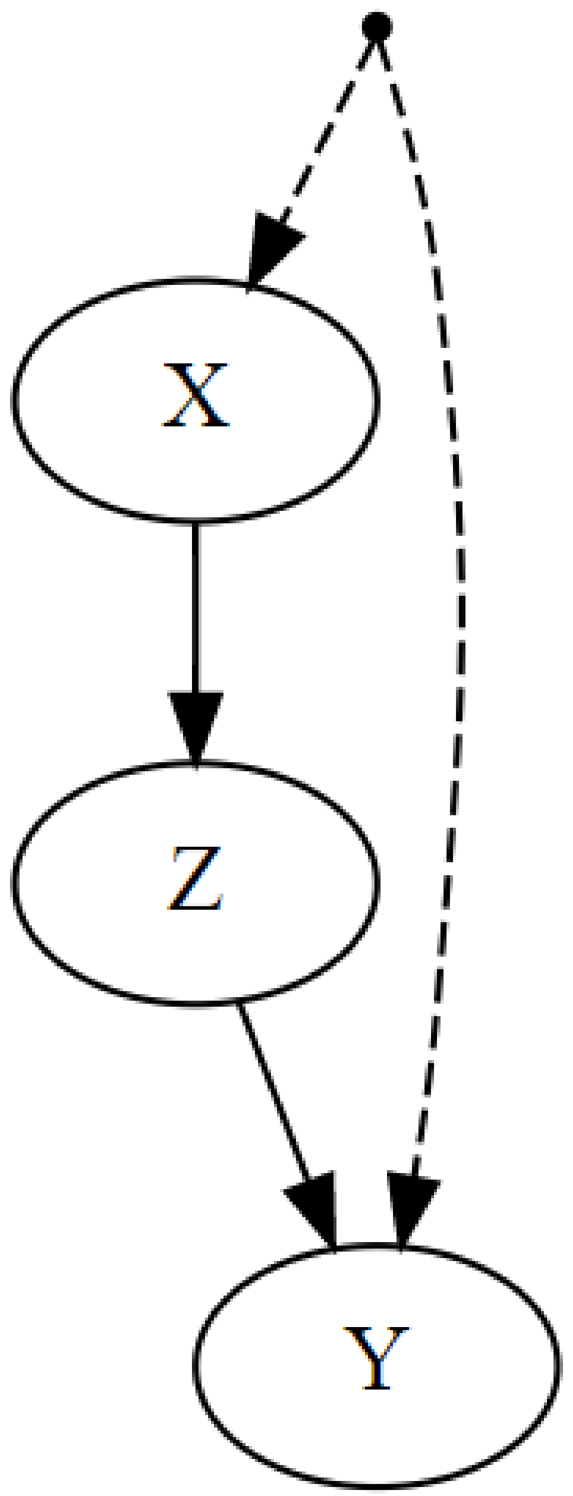
Let's consider the following smoking example:

- U (smoking gene) → unobserved confounders
- Z (tar deposit) → observed variables,
- X (smoking) causes Z (tar deposit)
- Z (tar deposit) outcomes Y (lung cancer)



The following method can be used to detect all the valid sets for applying the front-door criterion:

Unobserved confounders: Smoking example



Let's consider the following smoking example:

- U (smoking gene) → unobserved confounders
- Z (tar deposit) → observed variables,
- X (smoking) causes Z (tar deposit)
- Z (tar deposit) outcomes Y (lung cancer)

The following method can be used to detect all the valid sets for applying the front-door criterion:

```
bdnet3.get_all_frontdoor_adjustment_sets('X', 'Y')
```

```
frozenset({frozenset({'Z'})})
```

The answer is the intermediate variable Z as expected

Questions