# Inferencia causal mediante correlación sintáctica

Gabriel Goren


Tesis de Licenciatura

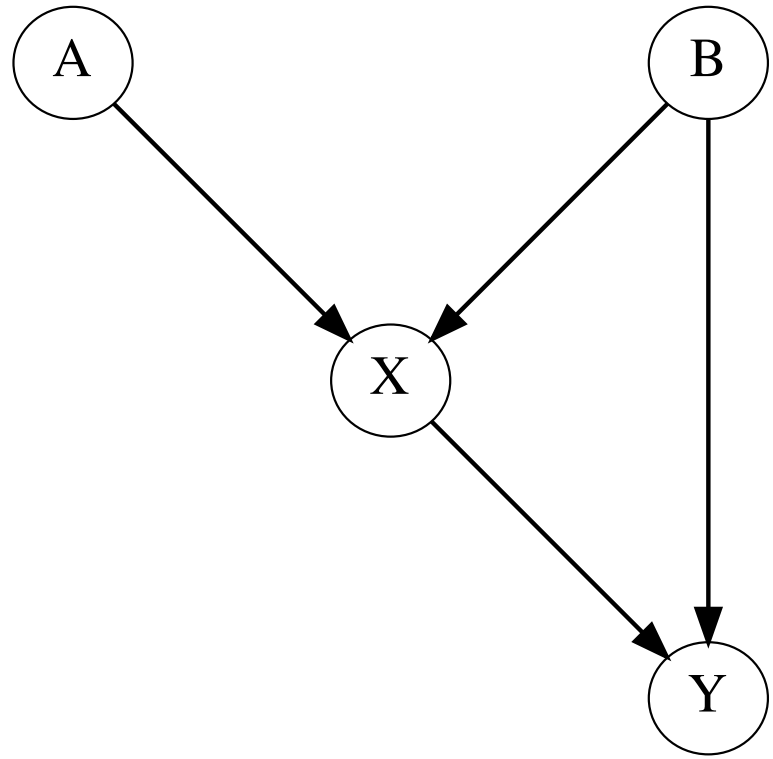
Director: Ariel Bendersky  Codirector: Santiago Figueira

Departamento de Física - Universidad de Buenos Aires

Jurado: Verónica Becher, Augusto Roncaglia, Enzo Tagliazucchi

26 de mayo de 2021

# Inferencia causal

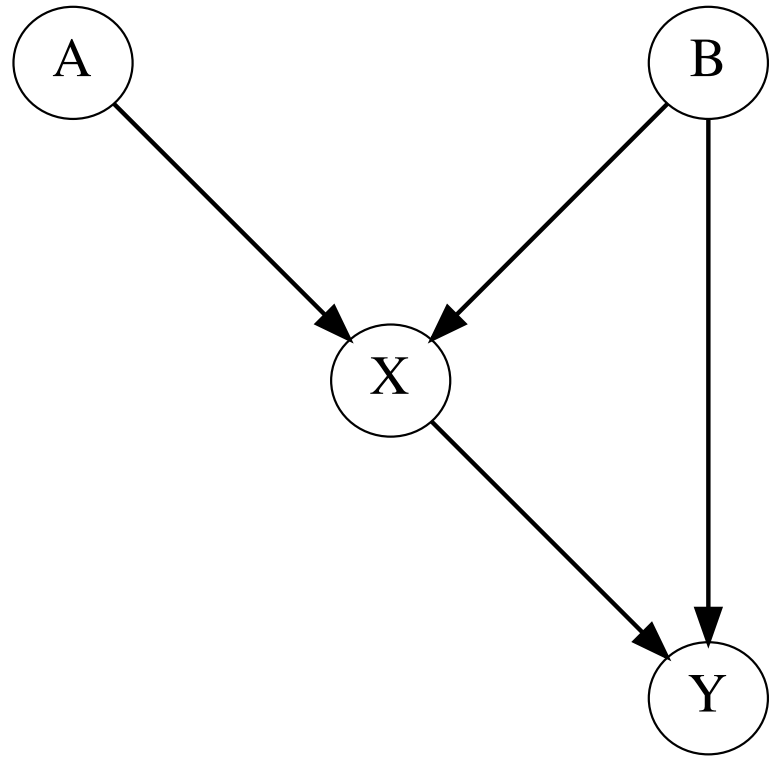



Un grafo acíclico dirigido (DAG)

# Inferencia causal

## Ejemplo

- $A$: Suministro eléctrico
- $B$: Condiciones climáticas
- $X$: Estado del sistema de riego
- $Y$: Humedad de la tierra

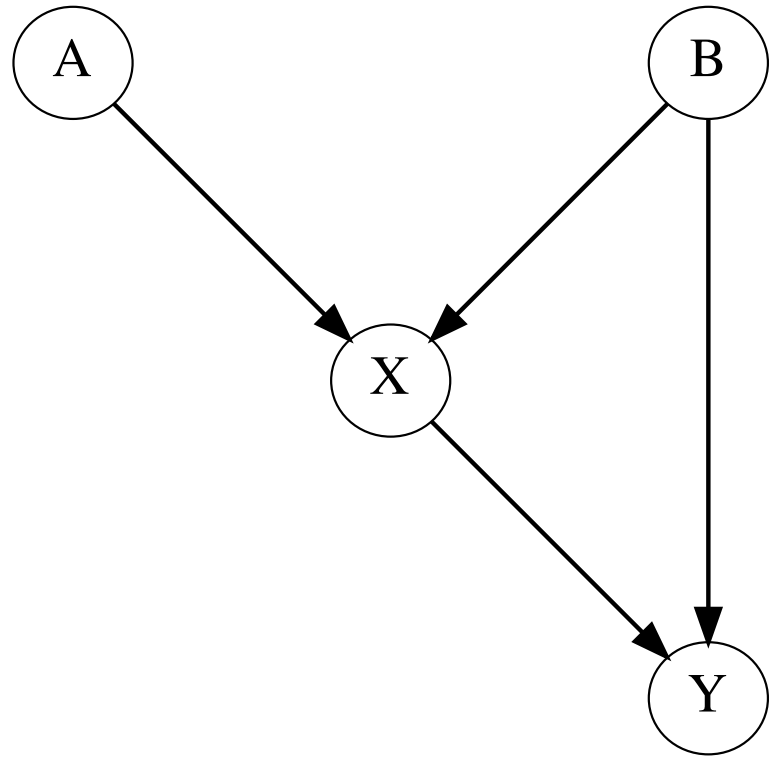



Un grafo acíclico dirigido (DAG)

# Inferencia causal

## Ejemplo

- $A$: Suministro eléctrico
- $B$: Condiciones climáticas
- $X$: Estado del sistema de riego
- $Y$: Humedad de la tierra



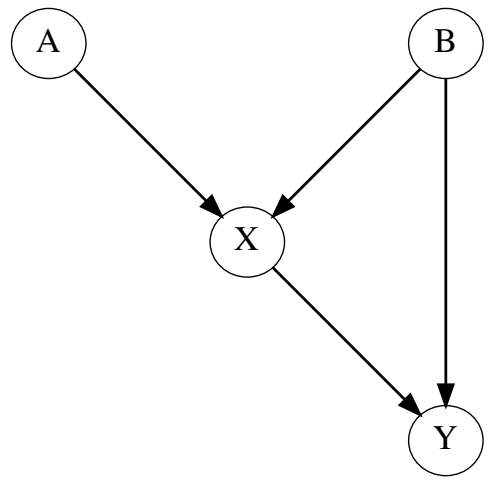

Un grafo acíclico dirigido (DAG)

Estamos interpretando las flechas como relaciones de *causalidad directa*.
Sin embargo, para formular esto con precisión debemos comenzar por una interpretación correlacional.

# Correlación estadística y DAGs

Supongamos que modelizamos a las variables de interés como variables aleatorias con distribución de probabilidad

$$p(a, b, x, y).$$




Entonces podemos leer el DAG como una *propuesta de factorización*:

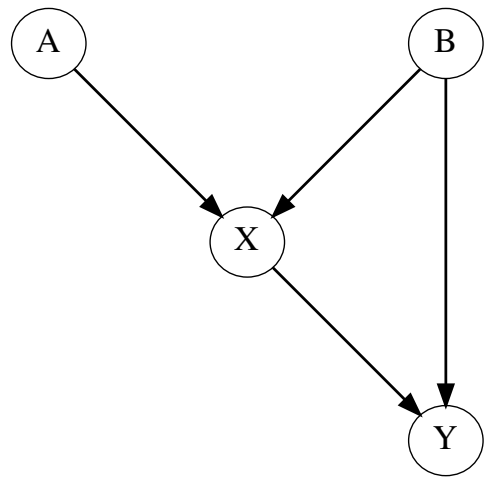
$$p(a, b, x, y) = p(y|x, b)\ p(x|a, b)\ p(a)\ p(b).$$

Judea Pearl. *Causality*. Cambridge university press, 2009.

# Correlación estadística y DAGs

Supongamos que modelizamos a las variables de interés como variables aleatorias con distribución de probabilidad

$$p(a, b, x, y).$$




Entonces podemos leer el DAG como una *propuesta de factorización*:

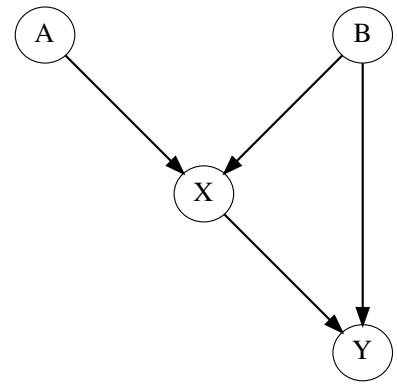
$$p(a, b, x, y) = p(y|x, b)\ p(x|a, b)\ p(a)\ p(b).$$

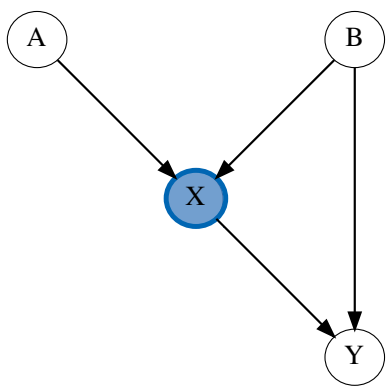
Si esta propuesta es válida, entonces el DAG es *Markov-compatible* con $p$.

Judea Pearl. *Causality*. Cambridge university press, 2009.

# *d-separación*

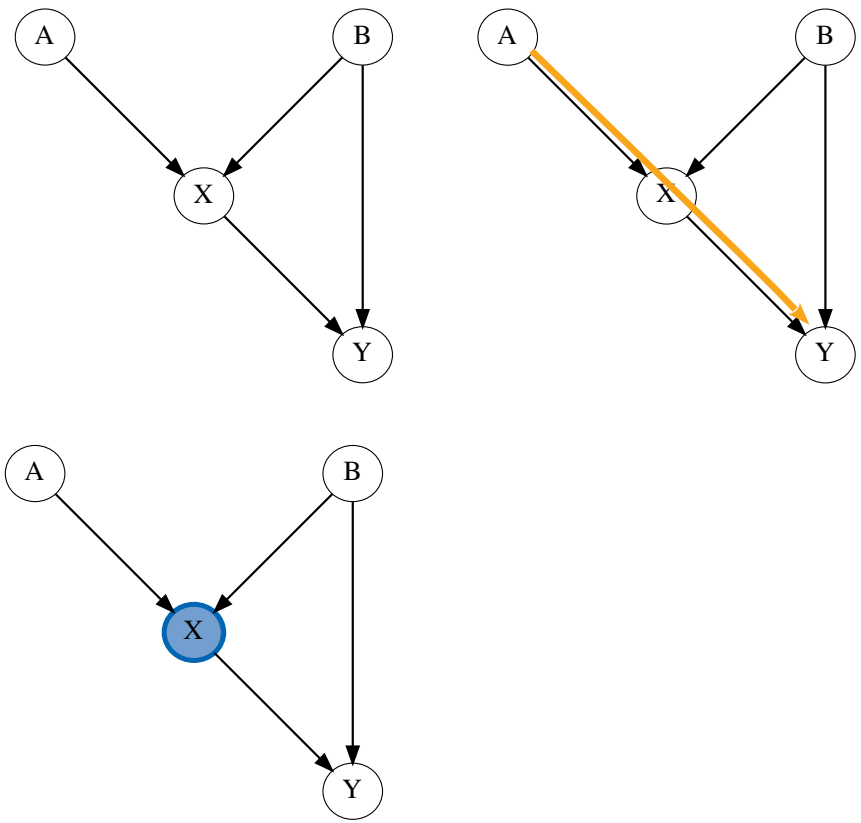
Podemos expresar una noción de independencia puramente gráfica, basada en la intuición de un flujo de información a lo largo del DAG:



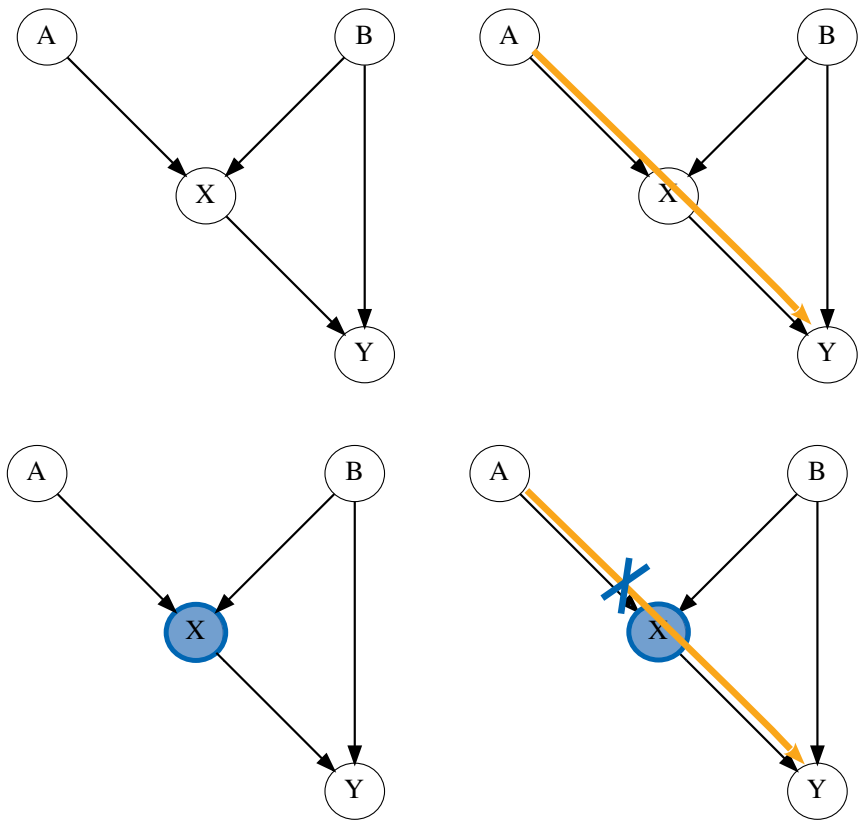
# *d-separación*

Podemos expresar una noción de independencia puramente gráfica, basada en la intuición de un flujo de información a lo largo del DAG:



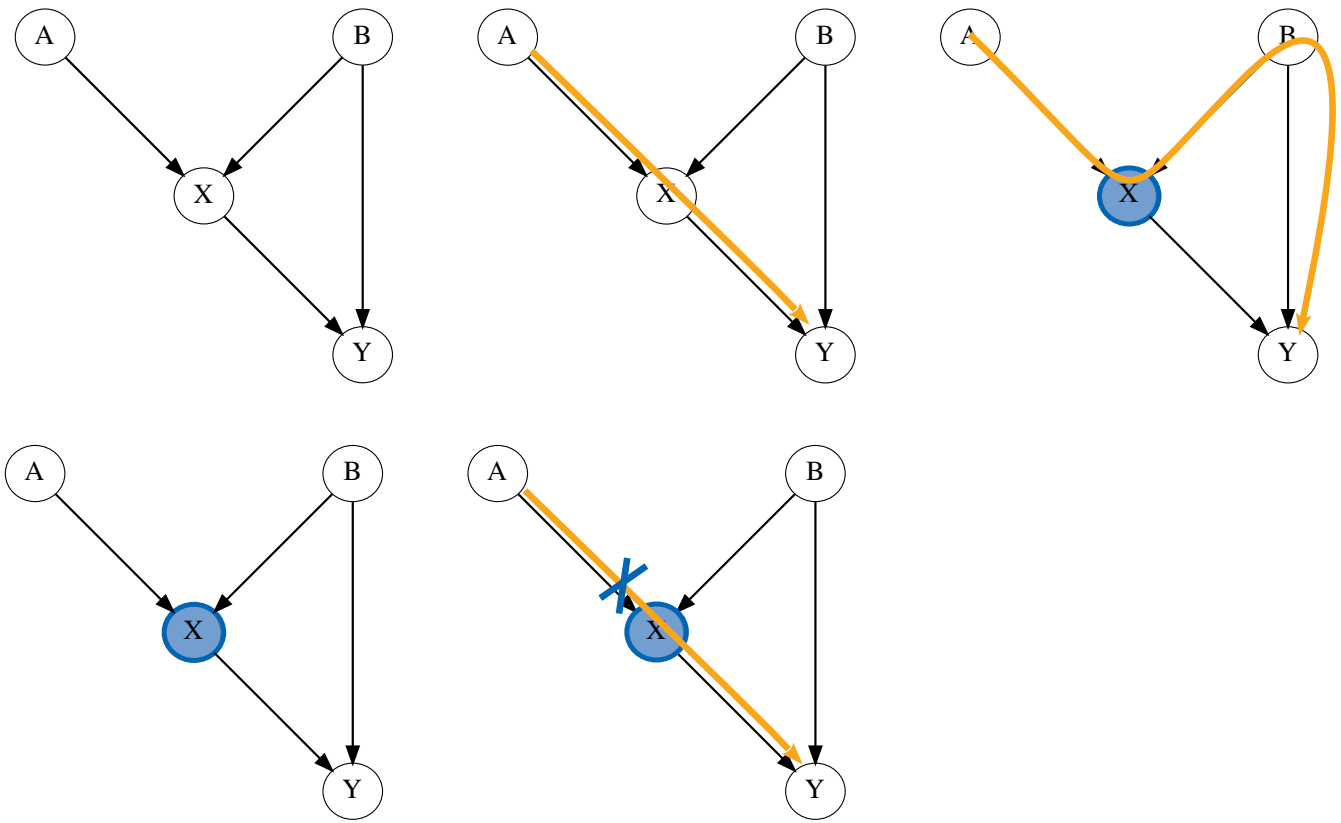
# *d*-separación

Podemos expresar una noción de independencia puramente gráfica, basada en la intuición de un flujo de información a lo largo del DAG:



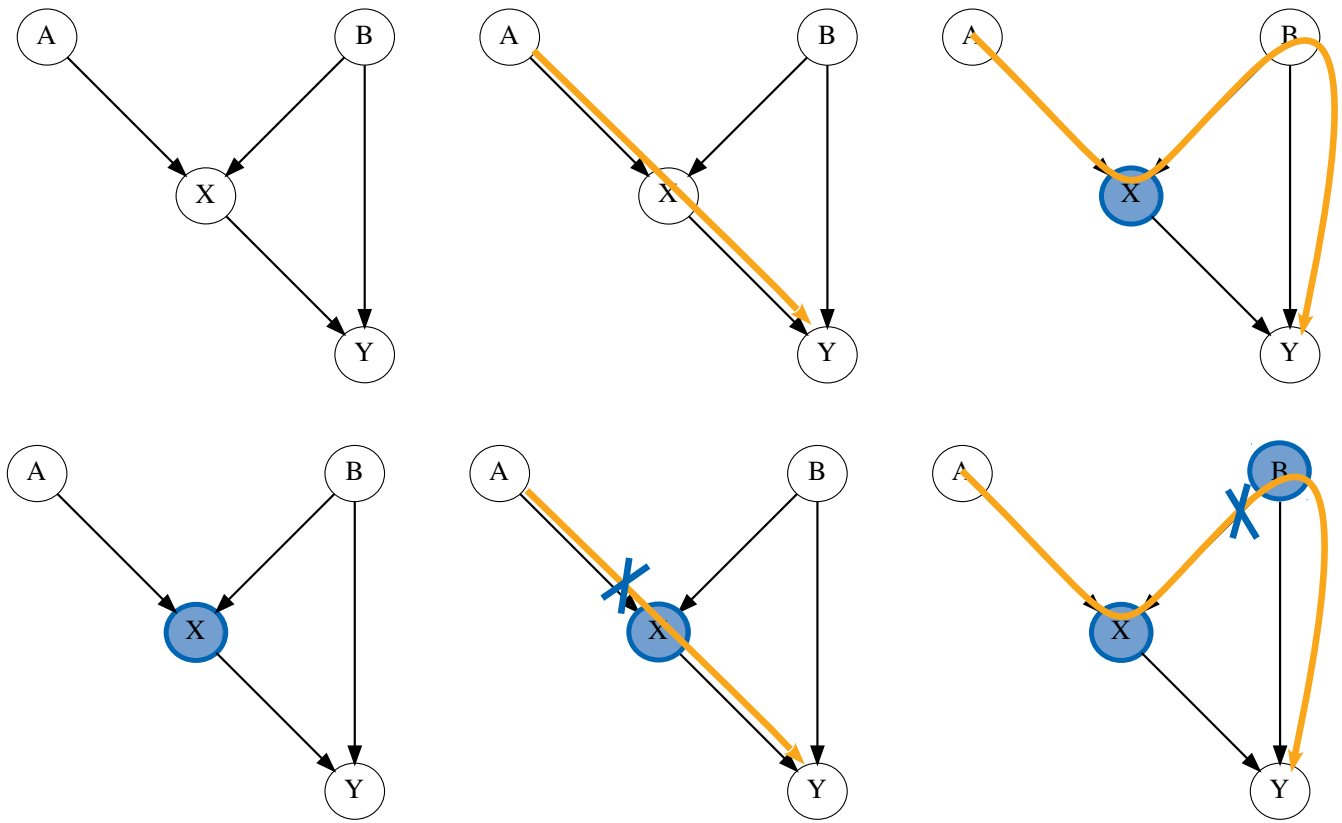
# *d-separación*

Podemos expresar una noción de independencia puramente gráfica, basada en la intuición de un flujo de información a lo largo del DAG:



# *d*-separación

Podemos expresar una noción de independencia puramente gráfica, basada en la intuición de un flujo de información a lo largo del DAG:



# d-separación

Como todos los caminos entre $A$ e $Y$ son *bloqueados* o *d-separados* por $\{X, B\}$, decimos que $A$ e $Y$ son independientes dados $X$ y $B$ con respecto al DAG:

$$(A \perp\!\!\!\perp Y | \{X, B\})_G .$$

# d-separación

Como todos los caminos entre $A$ e $Y$ son *bloqueados* o *d-separados* por $\{X, B\}$, decimos que $A$ e $Y$ son independientes dados $X$ y $B$ con respecto al DAG:

$$(A \perp\!\!\!\perp Y | \{X, B\})_G .$$

## Teorema

*$G$ es Markov-compatible con $p$ si y solo si d-separación implica independencia probabilística:*

$$(X \perp\!\!\!\perp Y | Z)_G \implies (X \perp\!\!\!\perp Y | Z)_p .$$

# Modelos causales funcionales

La manera más concreta de interpretar causalmente un DAG es a través de un modelo funcional.

# Modelos causales funcionales

La manera más concreta de interpretar causalmente un DAG es a través de un modelo funcional.

## Definición

Un *modelo causal funcional* o *estructural* consiste en

- Un DAG $G$ (*estructura causal* subyacente).
- Para cada nodo $j$, una variable aleatoria $U_j$ de *ruido*, todas independientes.
- Para cada nodo $j$, una *función estructural* $f_j$ que determina el valor de la variable $X_j$:

$$X_j = f_j(PA_j, U_j)$$

  donde $PA_j$ son los padres de $j$ en $G$.

Un modelo funcional induce una distribución de probabilidad sobre las variables $X_1, \ldots X_n$ (que es Markov-compatible con $G$).

# Descubrimiento causal

# Descubrimiento causal

Correlación no implica causalidad.

# Descubrimiento causal

Correlación no implica causalidad.
Causalidad sí implica correlación. ← hipótesis causal

# Descubrimiento causal

Correlación no implica causalidad.
Causalidad sí implica correlación. $\leftarrow$ hipótesis causal

Para inferir causalidad, necesitaremos hipótesis causales. Algunas de ellas:

# Descubrimiento causal

Correlación no implica causalidad.
Causalidad sí implica correlación. ← hipótesis causal

Para inferir causalidad, necesitaremos hipótesis causales. Algunas de ellas:

Ppio. de Reichenbach Si dos variables están correlacionadas, o bien una es causa de la otra o bien existe una tercera variable que es causa de las dos.

# Descubrimiento causal

Correlación no implica causalidad.
Causalidad sí implica correlación. ← hipótesis causal

Para inferir causalidad, necesitaremos hipótesis causales. Algunas de ellas:

**Ppio. de Reichenbach** Si dos variables están correlacionadas, o bien una es causa de la otra o bien existe una tercera variable que es causa de las dos.

**Estabilidad** Los datos no presentan independencias "espurias" que no estén reflejadas en la estructura causal (e.g. influencias causales que se cancelan entre sí).

(Otras hipótesis necesarias: *minimalidad*, *suficiencia causal*)

# Descubrimiento causal

Asumiendo estas hipótesis, tenemos el siguiente algoritmo.

- **Entrada:** una distribución $p$ sobre las variables $V$, proveniente de un modelo funcional desconocido, con estructura causal $G$.
- **Salida:** El esqueleto no dirigido de $G$.
- **El algoritmo**:
  - Para cada par de variables $(a, b)$ distintas en $V$, buscamos una manera de hacerlas independientes, i.e. un conjunto de variables $S_{ab}$ tal que $(a \perp\!\!\!\perp b | S_{ab})_p$.
  - Construimos un grafo $G'$ cuyos nodos son $V$, en el cual $a$ y $b$ están conectados si no hubo forma de hacerlas independientes, i.e. no se encontró ningún conjunto $S_{ab}$.

Tenemos entonces distintas formas de independencia condicional:

1. Independencia probabilística
2. Independencia gráfica / d-separación

Tenemos entonces distintas formas de independencia condicional:

1. Independencia probabilística
2. Independencia gráfica / d-separación
3. Independencia sintáctica / algorítmica

# Información algorítmica

- Ahora codificamos nuestras variables como *strings* (cadenas finitas de símbolos), e.g. $x = 011101011011$.

# Información algorítmica

- Ahora codificamos nuestras variables como *strings* (cadenas finitas de símbolos), e.g. $x = 011101011011$.
- Así podemos codificar distintos tipos de objetos (e.g. tablas de datos, textos, grafos).

# Información algorítmica

- Ahora codificamos nuestras variables como *strings* (cadenas finitas de símbolos), e.g. $x = 011101011011$.
- Así podemos codificar distintos tipos de objetos (e.g. tablas de datos, textos, grafos).
- Queremos una noción sintáctica de correlación entre objetos (strings), basada en su estructura compartida.

# Información algorítmica

- Ahora codificamos nuestras variables como *strings* (cadenas finitas de símbolos), e.g. $x = 011101011011$.
- Así podemos codificar distintos tipos de objetos (e.g. tablas de datos, textos, grafos).
- Queremos una noción sintáctica de correlación entre objetos (strings), basada en su estructura compartida.
- Para eso necesitamos cuantificar la estructura de cada objeto.

# Información algorítmica

Partimos de la siguiente intuición:

Un objeto *simple* es uno que puedo describir con pocas palabras.

Ming Li, Paul Vitányi, et al. *An introduction to Kolmogorov complexity and its applications*, volume 3. Springer, 2008.

# Información algorítmica

Partimos de la siguiente intuición:

Un objeto *simple* es uno que puedo describir con pocas palabras.

Ming Li, Paul Vitányi, et al. *An introduction to Kolmogorov complexity and its applications*, volume 3. Springer, 2008.

# Información algorítmica

Partimos de la siguiente intuición:

Un objeto *simple* es uno que puedo describir con pocas palabras.
Un string *simple* es uno que puedo generar con un programa corto.

Ming Li, Paul Vitányi, et al. *An introduction to Kolmogorov complexity and its applications*, volume 3. Springer, 2008.

# Información algorítmica

Partimos de la siguiente intuición:

Un objeto *simple* es uno que puedo describir con pocas palabras.
Un string *simple* es uno que puedo generar con un programa corto.

## Definición (Información algorítmica)

La *información algorítmica* (o *complejidad de Kolmogorov*) de un string $x$ se define como la longitud de un programa *minimal* que genera a $x$ como salida *(al ser ejecutado sin ninguna entrada)*:

$$K(x) = \min_p \{|p| : p \text{ devuelve } x \text{ al ser ejecutado}\}$$

(Para los detalles técnicos, ver Capítulo 3)

Idea: minimizar sobre todos los programas que calculan $x$ es encontrar la compresión computable óptima de $x$.

Ming Li, Paul Vitányi, et al. *An introduction to Kolmogorov complexity and its applications*, volume 3. Springer, 2008.

# Información algorítmica

## Ejemplo

Consideremos los siguientes strings binarios:

- $x_1 = 0101010101\ldots$ (01 repetido 10000 veces)
- $x_2 =$ capítulo 1 del Quijote en binario
- $x_3 =$ capítulos 1 y 2 del Quijote en binario comprimidos
- $x_4 =$ primeros 10000 digitos de $\pi$ en binario
- $x_5 = 0110111001\ldots$ (10000 tiradas de una moneda)

# Información algorítmica

## Ejemplo

Consideremos los siguientes strings binarios:

- $x_1 = 0101010101\ldots$ (01 repetido 10000 veces)
- $x_2 =$ capítulo 1 del Quijote en binario
- $x_3 =$ capítulos 1 y 2 del Quijote en binario comprimidos
- $x_4 =$ primeros 10000 digitos de $\pi$ en binario
- $x_5 = 0110111001\ldots$ (10000 tiradas de una moneda)

Intuitivamente,

- $x_2$ tiene más información que $x_1$
- $x_3$ tiene más información que $x_2$
- ¿$x_4$?

# Información algorítmica

## Ejemplo

Consideremos los siguientes strings binarios:

- $x_1 = 0101010101\ldots$ (01 repetido 10000 veces)
- $x_2 =$ capítulo 1 del Quijote en binario
- $x_3 =$ capítulos 1 y 2 del Quijote en binario comprimidos
- $x_4 =$ primeros 10000 digitos de $\pi$ en binario
- $x_5 = 0110111001\ldots$ (10000 tiradas de una moneda)

Intuitivamente,

- $x_2$ tiene más información que $x_1$
- $x_3$ tiene más información que $x_2$
- ¿$x_4$? Existen programas cortos para computar $\pi$, así que podría tener *menos* información que $x_2$ y $x_3$.

# Información algorítmica

## Ejemplo

Consideremos los siguientes strings binarios:

- $x_1 = 0101010101\ldots$ (01 repetido 10000 veces)
- $x_2 =$ capítulo 1 del Quijote en binario
- $x_3 =$ capítulos 1 y 2 del Quijote en binario comprimidos
- $x_4 =$ primeros 10000 digitos de $\pi$ en binario
- $x_5 = 0110111001\ldots$ (10000 tiradas de una moneda)

Intuitivamente,

- $x_2$ tiene más información que $x_1$
- $x_3$ tiene más información que $x_2$
- ¿$x_4$? Existen programas cortos para computar $\pi$, así que podría tener *menos* información que $x_2$ y $x_3$.
- $x_5$ no tiene regularidades, por lo que contiene *mucha* información.

# Información algorítmica: incomputabilidad

$K$ cuantifica *todos* los patrones computables en un objeto (incluso más allá de lo estadístico).

# Información algorítmica: incomputabilidad

$K$ cuantifica *todos* los patrones computables en un objeto (incluso más allá de lo estadístico).

Su potencia tiene una contracara:

**Teorema**

*La función* $K : \text{Strings} \to \mathbb{N}$ *es* incomputable*: no existe un programa que dado cualquier string* $x$ *como entrada devuelva* $K(x)$ *como salida.*

# Información algorítmica: incomputabilidad

$K$ cuantifica *todos* los patrones computables en un objeto (incluso más allá de lo estadístico).

Su potencia tiene una contracara:

**Teorema**
*La función* $K : \text{Strings} \to \mathbb{N}$ *es* incomputable*: no existe un programa que dado cualquier string* $x$ *como entrada devuelva* $K(x)$ *como salida.*

**Teorema**
*No existe ninguna función computable* $f : \text{Strings} \to \mathbb{N}$ *tal que* $|K(x) - f(x)| \leq c$ *para alguna constante* $c$ *y para todo* $x$*.*

# Información algorítmica: incomputabilidad

- La información algorítmica puede ser aproximada ejecutando cada vez más y más programas, pero no es posible controlar el error.
- También podemos aproximarla usando compresores comerciales (e.g. WinZip, gzip).
- Se trata de aproximaciones *por arriba*.

# Correlación algorítmica

# Correlación algorítmica

## Definición

La *información algorítmica condicional* de $x$ dado $y$ se define como

$$K(x|y) = \min_{p}\{|p| : p \text{ devuelve } x \text{ al ser ejecutado con entrada } y\}.$$

# Correlación algorítmica

## Definición

La *información algorítmica condicional* de $x$ dado $y$ se define como

$$K(x|y) = \min_{p}\{|p| : p \text{ devuelve } x \text{ al ser ejecutado con entrada } y\}.$$

Con esta variante condicional podemos definir la *información algorítmica mutua condicional*, que satisface

$$I(x : y|z) \simeq K(x|z) + K(y|z) - K(x, y|z)$$

# Correlación algorítmica

## Definición

La *información algorítmica condicional* de $x$ dado $y$ se define como

$$K(x|y) = \min_{p}\{|p| : p \text{ devuelve } x \text{ al ser ejecutado con entrada } y\}.$$

Con esta variante condicional podemos definir la *información algorítmica mutua condicional*, que satisface

$$I(x : y|z) \simeq K(x|z) + K(y|z) - K(x, y|z)$$

Definimos la relación de independencia condicional $(- \perp\!\!\!\perp - | -)_K$ según

$$(x \perp\!\!\!\perp y|z)_K \iff I(x : y|z) \approx 0$$

# Correlación algorítmica

## Ejemplo

Dado un string $x = x_1 x_2 \dots x_\ell$, su *shift* (o desplazamiento cíclico) *de magnitud* $k$ se define como

$$\text{shift}_k(x) := x_k x_{k+1} \dots x_\ell x_1 \dots x_{k_1}$$

# Correlación algorítmica

## Ejemplo

Dado un string $x = x_1 x_2 \dots x_\ell$, su *shift* (o desplazamiento cíclico) *de magnitud* $k$ se define como

$$\text{shift}_k(x) := x_k x_{k+1} \dots x_\ell x_1 \dots x_{k_1}$$

Tomemos

$$x = x_5 \qquad \text{(10000 tiradas de una moneda)}$$
$$y = \text{shift}_1(x)$$

Observando las frecuencias de aparición de 00, 01, 10, 11, no es posible concluir que $x$ e $y$ estén correlacionados.

# Correlación algorítmica

## Ejemplo

Dado un string $x = x_1 x_2 \dots x_\ell$, su *shift* (o desplazamiento cíclico) *de magnitud* $k$ se define como

$$\text{shift}_k(x) := x_k x_{k+1} \dots x_\ell x_1 \dots x_{k_1}$$

Tomemos

$$x = x_5 \qquad \text{(10000 tiradas de una moneda)}$$
$$y = \text{shift}_1(x)$$

Observando las frecuencias de aparición de 00, 01, 10, 11, no es posible concluir que $x$ e $y$ estén correlacionados.
Sin embargo existe un programa muy corto que computa la función $\text{shift}_1$, por lo cual $I(x : y) \gg 0$.

# Implementación

Se desarrolló una implementación en Python[1] del algoritmo de descubrimiento causal para operar sobre strings, junto con distintos tests de independencia.

- **Entrada:** Strings $x_1, \ldots x_n$ y un test de independencia condicional.

[1] `https://github.com/GRGab/tesislic-public`

# Implementación

Se desarrolló una implementación en Python[1] del algoritmo de descubrimiento causal para operar sobre strings, junto con distintos tests de independencia.

- **Entrada:** Strings $x_1, \ldots x_n$ y un test de independencia condicional.
- **Salida:** Un grafo no dirigido $G$ con vértices $x_1, \ldots x_n$.

[1] `https://github.com/GRGab/tesislic-public`

# Implementación

Se desarrolló una implementación en Python[1] del algoritmo de descubrimiento causal para operar sobre strings, junto con distintos tests de independencia.

- **Entrada:** Strings $x_1, \ldots x_n$ y un test de independencia condicional.
- **Salida:** Un grafo no dirigido $G$ con vértices $x_1, \ldots x_n$.
- **El algoritmo**:

[1] `https://github.com/GRGab/tesislic-public`

# Implementación

Se desarrolló una implementación en Python[1] del algoritmo de descubrimiento causal para operar sobre strings, junto con distintos tests de independencia.

- **Entrada:** Strings $x_1, \ldots x_n$ y un test de independencia condicional.
- **Salida:** Un grafo no dirigido $G$ con vértices $x_1, \ldots x_n$.
- **El algoritmo**:
  - Para cada par de variables $i, j$, buscamos un conjunto bloqueador $S_{ij}$ tal que $(x_i \perp\!\!\!\perp x_j | S_{ij})$.

[1] `https://github.com/GRGab/tesislic-public`

# Implementación

Se desarrolló una implementación en Python[1] del algoritmo de descubrimiento causal para operar sobre strings, junto con distintos tests de independencia.

- **Entrada:** Strings $x_1, \ldots x_n$ y un test de independencia condicional.
- **Salida:** Un grafo no dirigido $G$ con vértices $x_1, \ldots x_n$.
- **El algoritmo**:
  - Para cada par de variables $i, j$, buscamos un conjunto bloqueador $S_{ij}$ tal que $(x_i \perp\!\!\!\perp x_j | S_{ij})$.
  - Construimos un grafo $G$ en el cual $x_i$ y $x_j$ están conectados si no se encontró un conjunto bloqueador.

[1] `https://github.com/GRGab/tesislic-public`

# Tests de independencia

## Test estadístico

Tests $\chi^2$ combinados mediante un test KS. Considera a cada posición de los strings como una *ronda* independiente, por lo que ignora la información posicional en los strings.

# Tests de independencia

## Tests sintácticos

Dada una métrica de complejidad $Z$ (una aproximación a la información algorítmica) podemos definir una información mutua sintáctica:

$$Z(x|y) := Z(x,y) - Z(y)$$
$$I_Z(x:y|z) := Z(x|z) + Z(y|z) - Z(x,y|z).$$

Dada $Z$ y un valor de umbral $t > 0$, definimos

$$(x \perp\!\!\!\perp y | z_1, \ldots z_n)_{Z,t} \iff I_Z(x : y | \overbrace{\langle z_1, \ldots z_n \rangle}^{\text{concatenación}}) < t$$

En el rol de $Z$, fueron empleados el compresor comercial *gzip* y la *I-complejidad*[2].

[2] Verónica Becher y Pablo Ariel Heiber. A linearly computable measure of string complexity. *Theoretical Computer Science*, 438:62-73, 2012.

# Tests de independencia

Existen antecedentes teóricos que sugieren que este tipo de test sintáctico puede ser útil:

## Teorema (Compatibilidad de Markov algorítmica)

*Dado un grafo $G$ cuyos nodos corresponden a strings $x_1, \ldots x_n$, son equivalentes:*

1. *Cada $x_i$ es algorítmicamente independiente de todos sus no-descendientes, dados sus padres según $G$.*
2. $(X \perp\!\!\!\perp Y|Z)_G \implies (X \perp\!\!\!\perp Y|Z)_K$

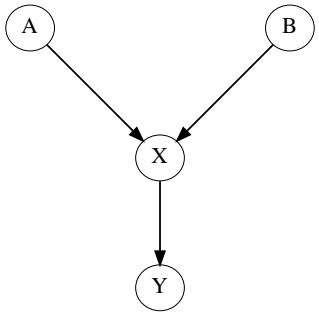
También hay una noción de *modelo funcional algorítmico* asociado.

Dominik Janzing y Bernhard Schölkopf. *Causal inference using the algorithmic markov condition*. IEEE Transactions on Information Theory, 56(10):5168-5194,2010.

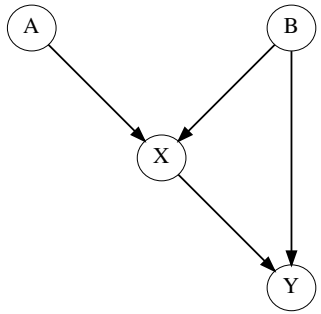
# Modelos generadores

Para testear el algoritmo, se generaron casos de prueba empleando ciertos *modelos generadores*.

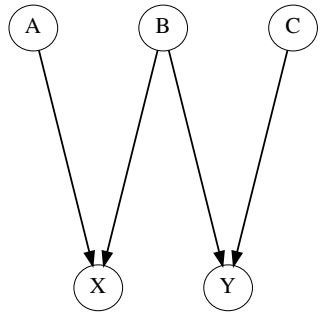
## Estructuras causales



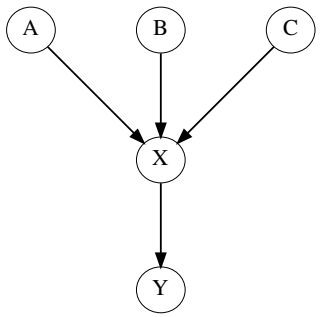

(a) Grafo Y



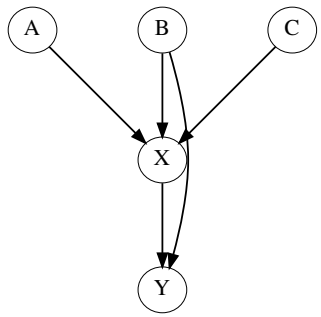

(b) Grafo instrumental




(c) Grafo W




(d) Grafo Ψ




(e) Grafo Ψ instrumental

# Modelos generadores: funciones estructurales

# Modelos generadores: funciones estructurales

1. **Funciones tipo XOR**: $s_1, \ldots s_k \mapsto s_1 \oplus \ldots \oplus s_k$

$$
\begin{array}{r}
0010110 \\
\oplus 0000111 \\
= 0010001
\end{array}
$$

# Modelos generadores: funciones estructurales

1. **Funciones tipo XOR**: $s_1, \ldots s_k \mapsto s_1 \oplus \ldots \oplus s_k$

$$
\begin{aligned}
&0010110\\
\oplus&0000111\\
=&0010001
\end{aligned}
$$

2. **Funciones de concatenación truncada:**

$$
\begin{aligned}
&f\left(a_1 a_2 a_3 a_4 a_5 a_6, b_1 b_2 b_3 b_4 b_5 b_6,\right.\\
&\left.c_1 c_2 c_3 c_4 c_5 c_6\right)\\
=&a_1 a_2 b_3 b_3 c_5 c_6
\end{aligned}
$$

# Modelos generadores: funciones estructurales

1. **Funciones tipo XOR**: $s_1, \ldots s_k \mapsto s_1 \oplus \ldots \oplus s_k$

$$
\begin{aligned}
&0010110\\
\oplus&0000111\\
=&0010001
\end{aligned}
$$

2. **Funciones de concatenación truncada:**

$$
\begin{aligned}
&f\left(a_1\, a_2\, a_3\, a_4\, a_5\, a_6\,,\, b_1\, b_2\, b_3\, b_4\, b_5\, b_6\,,\right.\\
&\left.\quad c_1\, c_2\, c_3\, c_4\, c_5\, c_6\right)\\
=&a_1\, a_2\, b_3\, b_3\, c_5\, c_6
\end{aligned}
$$

3. **Funciones de concatenación errática:** se determina aleatoriamente la longitud de los fragmentos que son copiados, y de qué argumento provienen.

# Modelos generadores: funciones estructurales

- Se utilizaron strings binarios de longitud $\ell = 20000$.

# Modelos generadores: funciones estructurales

- Se utilizaron strings binarios de longitud $\ell = 20000$.
- En los nodos raíz, se generaron strings aleatorios con probabilidad $p_1 = 0{,}4$ de que el símbolo en una dada posición sea 1.

# Modelos generadores: funciones estructurales

- Se utilizaron strings binarios de longitud $\ell = 20000$.
- En los nodos raíz, se generaron strings aleatorios con probabilidad $p_1 = 0{,}4$ de que el símbolo en una dada posición sea 1.
- Se agregó ruido a todas las funciones estructurales, con una probabilidad $p_{\text{ruido}} = 0{,}1$ de que un dado bit en el resultado de la función sea invertido.

# Modelos generadores: funciones estructurales

- Se utilizaron strings binarios de longitud $\ell = 20000$.
- En los nodos raíz, se generaron strings aleatorios con probabilidad $p_1 = 0{,}4$ de que el símbolo en una dada posición sea 1.
- Se agregó ruido a todas las funciones estructurales, con una probabilidad $p_{\text{ruido}} = 0{,}1$ de que un dado bit en el resultado de la función sea invertido.
- **Modelos con shifts:** Para cada una de las 3 familias de funciones, se estudió una variante en la que se introducen shifts en los nodos $X$ e $Y$, con magnitudes diferentes.
  Esto busca romper la correlación ronda a ronda entre los strings.

# Generación de casos de prueba

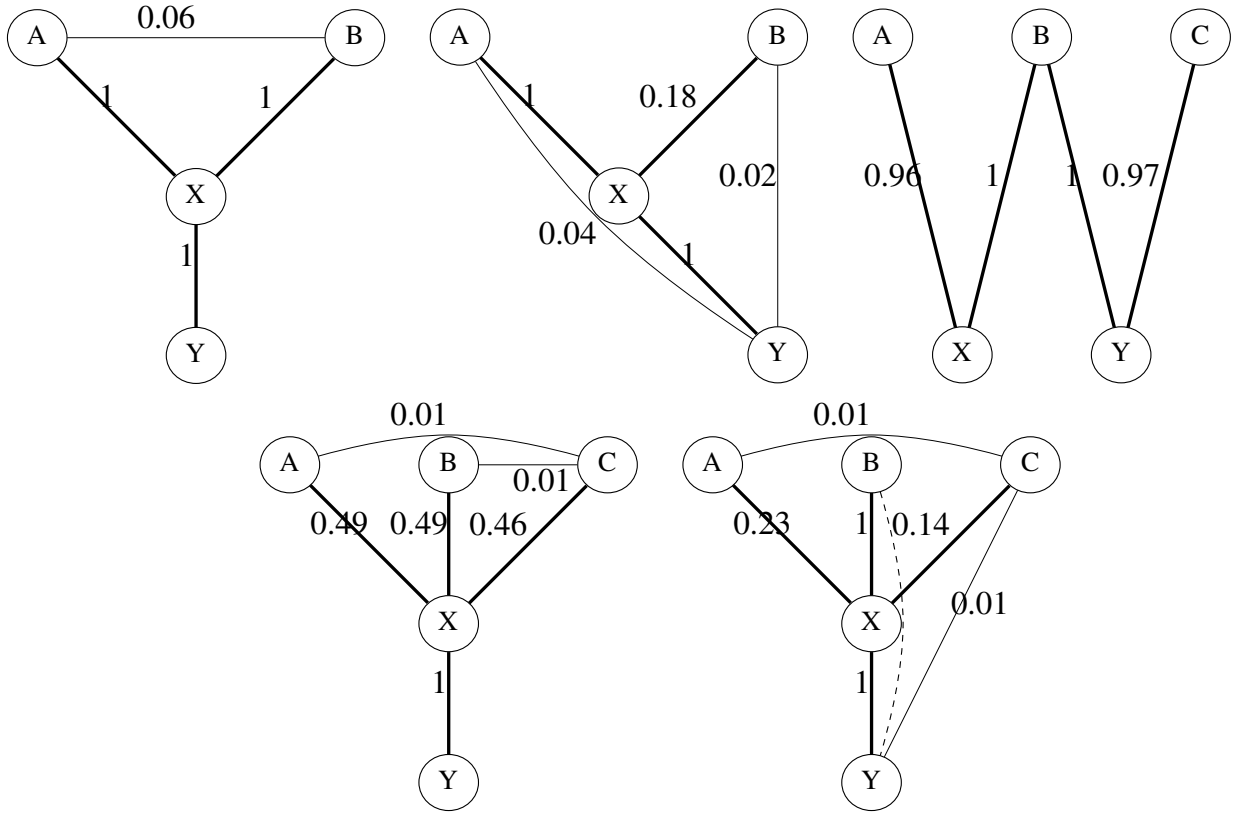
- Para cada elección de modelo generador y de test de independencia, se ejecutó el algoritmo sobre 100 conjuntos de strings diferentes generados aleatoriamente.

# Generación de casos de prueba

- Para cada elección de modelo generador y de test de independencia, se ejecutó el algoritmo sobre 100 conjuntos de strings diferentes generados aleatoriamente.
- Como resultado, obtenemos una frecuencia de aparición para cada arista posible.

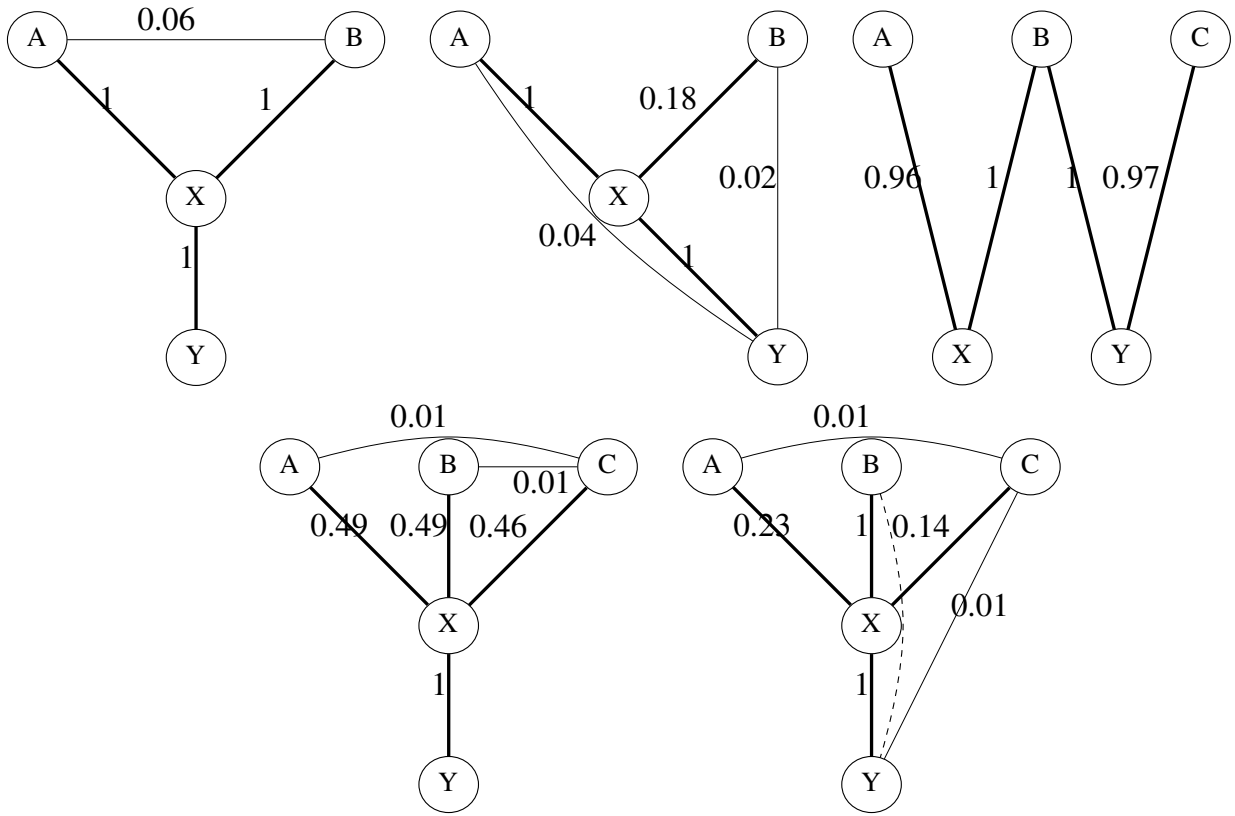
# Resultados: test estadístico

## Modelos XOR



# Resultados: test estadístico

## Modelos XOR

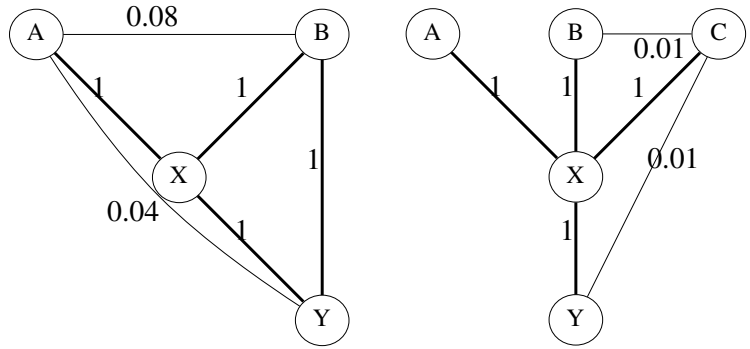



- En líneas generales se infieren las aristas del grafo subyacente.
- Sin embargo, ciertas combinaciones de estructura causal y funciones estructurales son difíciles (o imposibles) de inferir.

# Resultados: test estadístico

## Modelos de concatenación

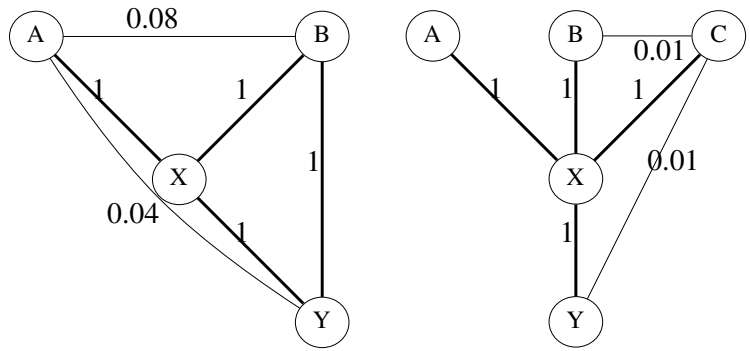
- En líneas generales se infieren las aristas del grafo subyacente.
- Ejemplo: concatenación truncada



# Resultados: test estadístico

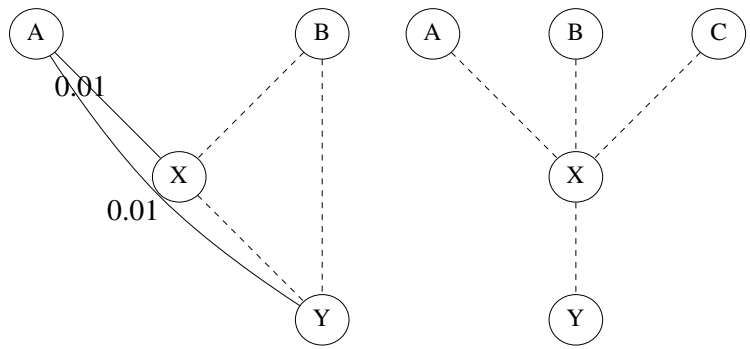
## Modelos de concatenación

- En líneas generales se infieren las aristas del grafo subyacente.
- Ejemplo: concatenación truncada




## Modelos con shifts

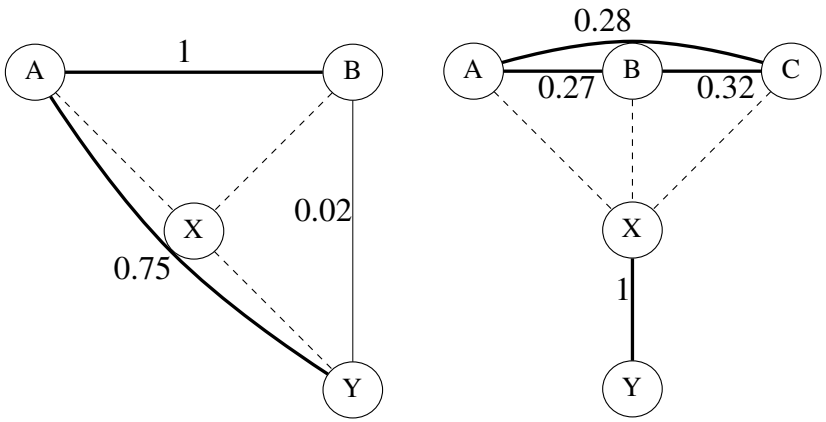
- El test estadístico *no* es capaz de detectar correlaciones.



# Resultados: test gzip

## Modelos XOR

- El test sintáctico *no* es capaz de detectar la mayoría de las aristas usando $t = 0$.

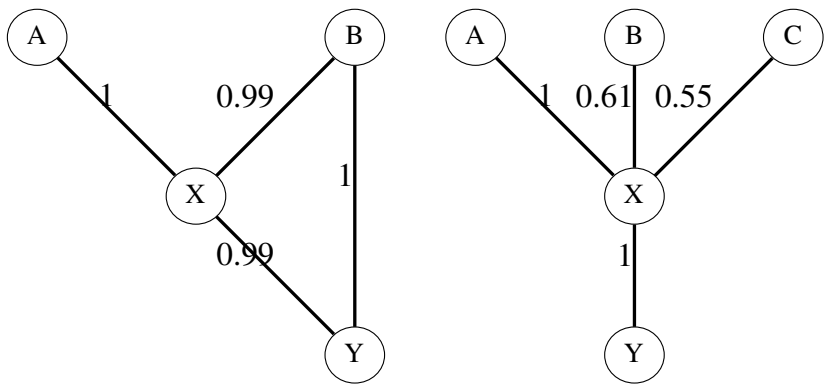



- Como hay tanto falsos negativos como falsos positivos, podemos concluir que no existe una elección de $t$ tal que el test infiera los grafos subyacentes.

# Resultados: test gzip

## Modelos de concatenación, con y sin shifts

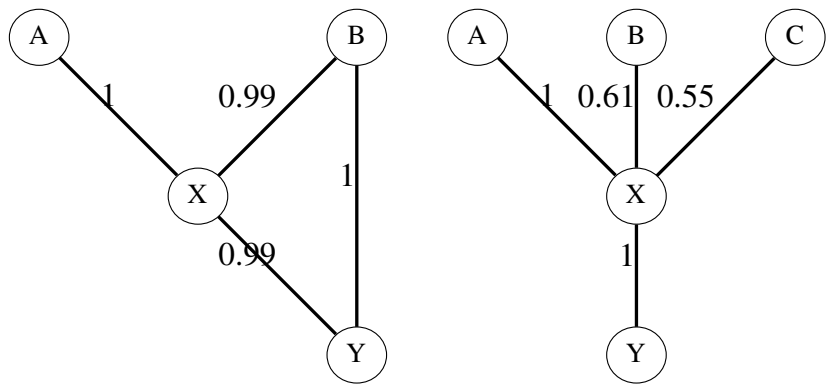
- Empleando $t = 0$, los grafos subyacentes son inferidos la mayoría de las veces en todos los casos, en general con probabilidades $\sim 0{,}9$.
- Ejemplo: concatenación truncada



# Resultados: test gzip

## Modelos de concatenación, con y sin shifts

- Empleando $t = 0$, los grafos subyacentes son inferidos la mayoría de las veces en todos los casos, en general con probabilidades $\sim 0{,}9$.
- Ejemplo: concatenación truncada



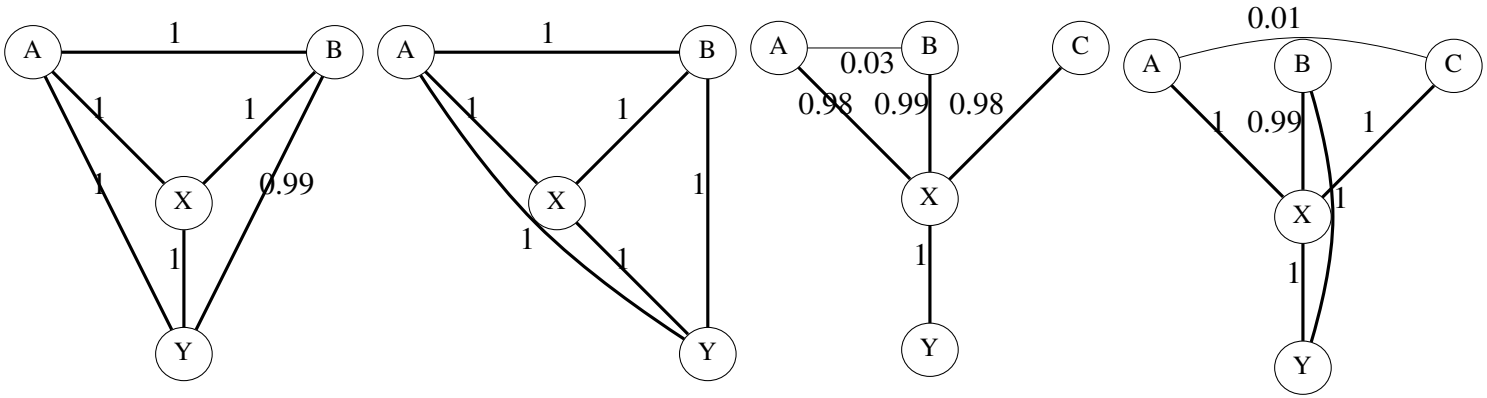

- No se observan diferencias notables entre los casos con y sin shifts.

# Resultados: test l-complejidad

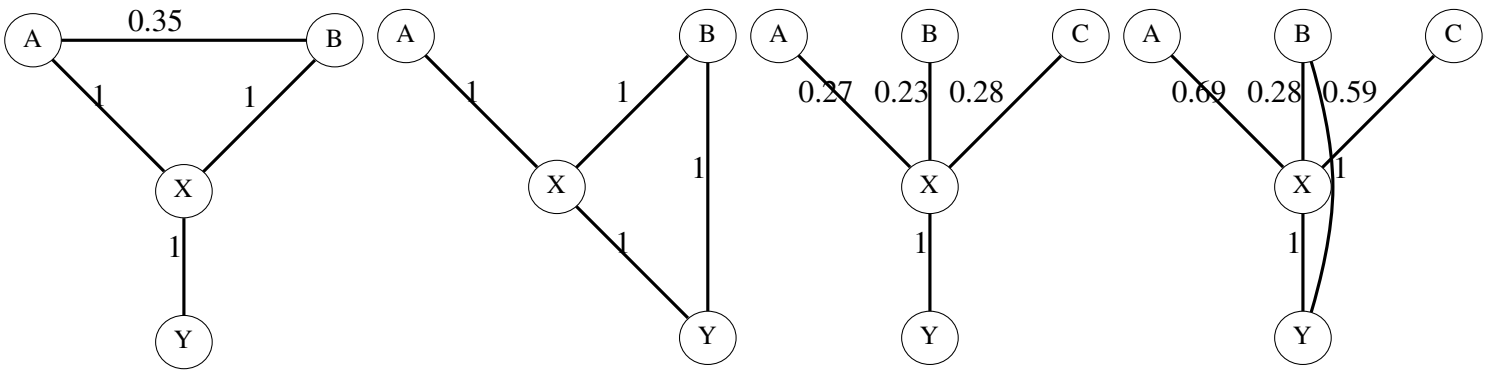
- A grandes rasgos el comportamiento es análogo al del test gzip.
- Sin embargo, no existe un umbral $t$ tal que todos las estructuras causales sean inferidas.

# Resultados: test l-complejidad

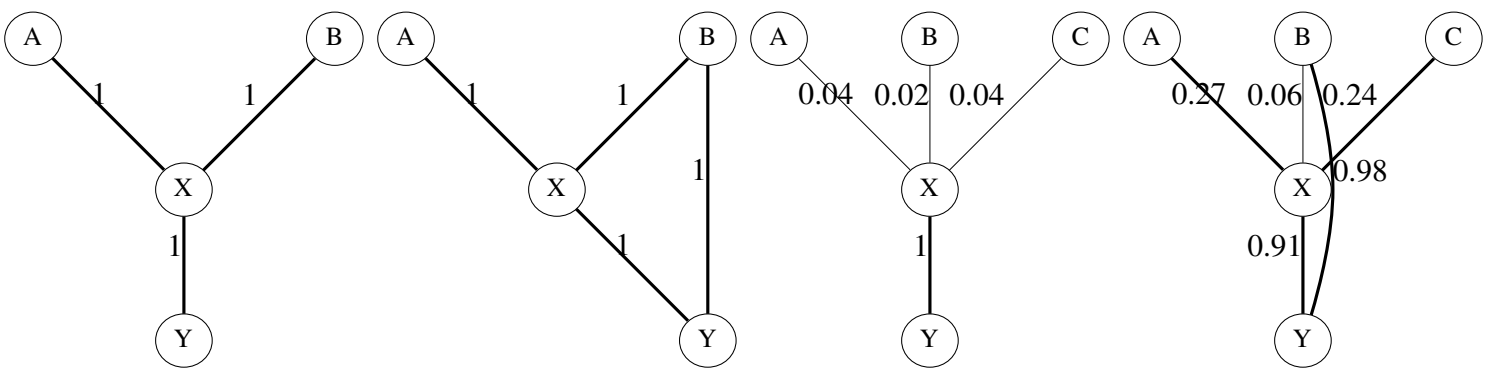
$t = 35$




$t = 45$




$t = 50$



# Discusión

# Discusión

- Como era de esperar, el test estadístico detecta correlaciones ronda a ronda (XOR), y no detecta correlaciones ante la presencia de shifts.
- Inversamente, se observó que los tests sintácticos
  - tienen dificultades para identificar correlaciones ronda a ronda;
  - sin embargo, detectan las correlaciones sin problemas ante la presencia de shifts.

# Discusión

- Como era de esperar, el test estadístico detecta correlaciones ronda a ronda (XOR), y no detecta correlaciones ante la presencia de shifts.
- Inversamente, se observó que los tests sintácticos
  - tienen dificultades para identificar correlaciones ronda a ronda;
  - sin embargo, detectan las correlaciones sin problemas ante la presencia de shifts.
- Se observó que *condicionar sobre más strings siempre baja la información mutua sintáctica* $I_Z$. Esto implica un criterio simple para determinar si una arista va a ser inferida por el algoritmo:
  Una arista $(V_i, V_j)$ es inferida a partir de una colección de strings de entrada $v_1 \dots v_n$ si y solo si

$$t < t_{ij} := I_Z(v_i : v_j | \langle \{v_k : k \neq i, j\} \rangle)$$

# Discusión

- Más aún, se observó que los valores $t_{ij}$ caen sistemáticamente por debajo del valor de $I_Z(a : b)$.

# Discusión

- Más aún, se observó que los valores $t_{ij}$ caen sistemáticamente por debajo del valor de $I_Z(a : b)$.
- Por lo tanto, cuando se concluye que no hay una arista entre $A$ y $B$, se está empleando un conjunto bloqueador no vacío.

# Discusión

- Más aún, se observó que los valores $t_{ij}$ caen sistemáticamente por debajo del valor de $I_Z(a : b)$.
- Por lo tanto, cuando se concluye que no hay una arista entre $A$ y $B$, se está empleando un conjunto bloqueador no vacío.
- Es decir que los conjuntos bloqueadores hallados no son siempre los "correctos" ($A$ y $B$ deberían ser *marginalmente* independientes).

# Discusión

- Más aún, se observó que los valores $t_{ij}$ caen sistemáticamente por debajo del valor de $I_Z(a : b)$.
- Por lo tanto, cuando se concluye que no hay una arista entre $A$ y $B$, se está empleando un conjunto bloqueador no vacío.
- Es decir que los conjuntos bloqueadores hallados no son siempre los "correctos" ($A$ y $B$ deberían ser *marginalmente* independientes).
- Esto no es relevante para el algoritmo implementado, pero sí para extensiones del mismo que buscan inferir las direcciones de (algunas) aristas.

# Perspectivas y trabajo a futuro

[3] Bastian Steudel, Dominik Janzing, y Bernhard Schölkopf. *Causal markov condition for submodular information measures*. arXiv preprint arXiv:1002.4020, 2010.

# Perspectivas y trabajo a futuro

- Estudiar otros tipos de funciones estructurales y transformaciones sintácticas de strings.

[3] Bastian Steudel, Dominik Janzing, y Bernhard Schölkopf. *Causal markov condition for submodular information measures*. arXiv preprint arXiv:1002.4020, 2010.

# Perspectivas y trabajo a futuro

- Estudiar otros tipos de funciones estructurales y transformaciones sintácticas de strings.
  - Descubrimiento causal como estándar para evaluar la adecuación de una aproximacion a la información algorítmica con respecto a una clase de procesos causales.

[3] Bastian Steudel, Dominik Janzing, y Bernhard Schölkopf. *Causal markov condition for submodular information measures*. arXiv preprint arXiv:1002.4020, 2010.

# Perspectivas y trabajo a futuro

- Estudiar otros tipos de funciones estructurales y transformaciones sintácticas de strings.
  - Descubrimiento causal como estándar para evaluar la adecuación de una aproximacion a la información algorítmica con respecto a una clase de procesos causales.
  - Caracterizar teóricamente la relación entre funciones estructurales y métricas de complejidad[3].

[3] Bastian Steudel, Dominik Janzing, y Bernhard Schölkopf. *Causal markov condition for submodular information measures*. arXiv preprint arXiv:1002.4020, 2010.

# Perspectivas y trabajo a futuro

- Estudiar otros tipos de funciones estructurales y transformaciones sintácticas de strings.
  - Descubrimiento causal como estándar para evaluar la adecuación de una aproximacion a la información algorítmica con respecto a una clase de procesos causales.
  - Caracterizar teóricamente la relación entre funciones estructurales y métricas de complejidad[3].
- Estudiar extensiones del algoritmo.

[3] Bastian Steudel, Dominik Janzing, y Bernhard Schölkopf. *Causal markov condition for submodular information measures*. arXiv preprint arXiv:1002.4020, 2010.

# Perspectivas y trabajo a futuro

- Estudiar otros tipos de funciones estructurales y transformaciones sintácticas de strings.
  - Descubrimiento causal como estándar para evaluar la adecuación de una aproximacion a la información algorítmica con respecto a una clase de procesos causales.
  - Caracterizar teóricamente la relación entre funciones estructurales y métricas de complejidad[3].
- Estudiar extensiones del algoritmo.
  - Fases de direccionamiento de aristas.

[3] Bastian Steudel, Dominik Janzing, y Bernhard Schölkopf. *Causal markov condition for submodular information measures*. arXiv preprint arXiv:1002.4020, 2010.

# Perspectivas y trabajo a futuro

- Estudiar otros tipos de funciones estructurales y transformaciones sintácticas de strings.
  - Descubrimiento causal como estándar para evaluar la adecuación de una aproximacion a la información algorítmica con respecto a una clase de procesos causales.
  - Caracterizar teóricamente la relación entre funciones estructurales y métricas de complejidad[3].
- Estudiar extensiones del algoritmo.
  - Fases de direccionamiento de aristas.
  - Tests de independencia condicional *difusos* que no utilicen un único valor de $t$.

[3] Bastian Steudel, Dominik Janzing, y Bernhard Schölkopf. *Causal markov condition for submodular information measures*. arXiv preprint arXiv:1002.4020, 2010.

# Perspectivas y trabajo a futuro

- Estudiar otros tipos de funciones estructurales y transformaciones sintácticas de strings.
  - Descubrimiento causal como estándar para evaluar la adecuación de una aproximacion a la información algorítmica con respecto a una clase de procesos causales.
  - Caracterizar teóricamente la relación entre funciones estructurales y métricas de complejidad[3].
- Estudiar extensiones del algoritmo.
  - Fases de direccionamiento de aristas.
  - Tests de independencia condicional *difusos* que no utilicen un único valor de $t$.
- Entender en profundidad las causas de que las estructuras causales solo puedan ser reconstruidas con valores "demasiado bajos" de $t$.

[3] Bastian Steudel, Dominik Janzing, y Bernhard Schölkopf. *Causal markov condition for submodular information measures*. arXiv preprint arXiv:1002.4020, 2010.

# Perspectivas y trabajo a futuro

- Estudiar otros tipos de funciones estructurales y transformaciones sintácticas de strings.
  - Descubrimiento causal como estándar para evaluar la adecuación de una aproximacion a la información algorítmica con respecto a una clase de procesos causales.
  - Caracterizar teóricamente la relación entre funciones estructurales y métricas de complejidad[3].
- Estudiar extensiones del algoritmo.
  - Fases de direccionamiento de aristas.
  - Tests de independencia condicional *difusos* que no utilicen un único valor de $t$.
- Entender en profundidad las causas de que las estructuras causales solo puedan ser reconstruidas con valores "demasiado bajos" de $t$.
- Aplicaciones de la metodología desarrollada a dominios específicos.

[3] Bastian Steudel, Dominik Janzing, y Bernhard Schölkopf. *Causal markov condition for submodular information measures*. arXiv preprint arXiv:1002.4020, 2010.

# Resumen y conclusiones

- Se desarrolló un abordaje a la inferencia de modelos causales basado en nociones sintácticas de correlación.

# Resumen y conclusiones

- Se desarrolló un abordaje a la inferencia de modelos causales basado en nociones sintácticas de correlación.
- Se implementó un algoritmo para el descubrimiento del esqueleto causal de un conjunto de datos simbólicos y se desarrollaron tests de independencia estadísticos y sintácticos.

# Resumen y conclusiones

- Se desarrolló un abordaje a la inferencia de modelos causales basado en nociones sintácticas de correlación.
- Se implementó un algoritmo para el descubrimiento del esqueleto causal de un conjunto de datos simbólicos y se desarrollaron tests de independencia estadísticos y sintácticos.
- Se construyeron ciertos modelos funcionales con los cuales se generaron casos de prueba.

# Resumen y conclusiones

- Se desarrolló un abordaje a la inferencia de modelos causales basado en nociones sintácticas de correlación.
- Se implementó un algoritmo para el descubrimiento del esqueleto causal de un conjunto de datos simbólicos y se desarrollaron tests de independencia estadísticos y sintácticos.
- Se construyeron ciertos modelos funcionales con los cuales se generaron casos de prueba.
- Fueron identificadas clases de modelos cuyos esqueletos causales

# Resumen y conclusiones

- Se desarrolló un abordaje a la inferencia de modelos causales basado en nociones sintácticas de correlación.
- Se implementó un algoritmo para el descubrimiento del esqueleto causal de un conjunto de datos simbólicos y se desarrollaron tests de independencia estadísticos y sintácticos.
- Se construyeron ciertos modelos funcionales con los cuales se generaron casos de prueba.
- Fueron identificadas clases de modelos cuyos esqueletos causales
  1. pueden ser inferidos empleando tests estadísticos, mas no con tests sintácticos (modelos tipo XOR);

# Resumen y conclusiones

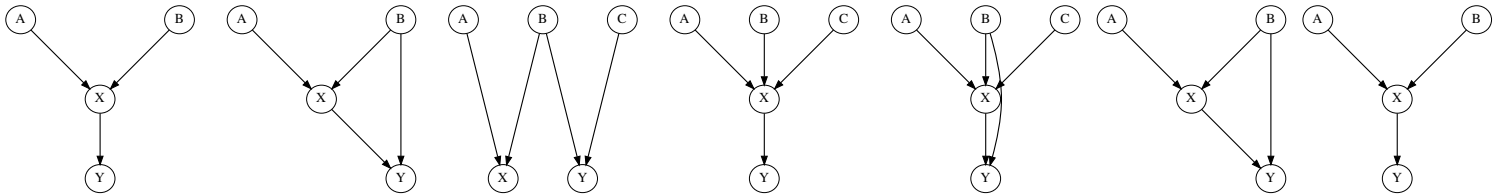
- Se desarrolló un abordaje a la inferencia de modelos causales basado en nociones sintácticas de correlación.
- Se implementó un algoritmo para el descubrimiento del esqueleto causal de un conjunto de datos simbólicos y se desarrollaron tests de independencia estadísticos y sintácticos.
- Se construyeron ciertos modelos funcionales con los cuales se generaron casos de prueba.
- Fueron identificadas clases de modelos cuyos esqueletos causales
  1. pueden ser inferidos empleando tests estadísticos, mas no con tests sintácticos (modelos tipo XOR);
  2. pueden ser reconstruidos mediante ambos tipos de test (modelos de concatenación); y

# Resumen y conclusiones

- Se desarrolló un abordaje a la inferencia de modelos causales basado en nociones sintácticas de correlación.
- Se implementó un algoritmo para el descubrimiento del esqueleto causal de un conjunto de datos simbólicos y se desarrollaron tests de independencia estadísticos y sintácticos.
- Se construyeron ciertos modelos funcionales con los cuales se generaron casos de prueba.
- Fueron identificadas clases de modelos cuyos esqueletos causales
  1. pueden ser inferidos empleando tests estadísticos, mas no con tests sintácticos (modelos tipo XOR);
  2. pueden ser reconstruidos mediante ambos tipos de test (modelos de concatenación); y
  3. solo pueden ser reconstruidos mediante tests sintácticos (modelos de concatenación con shifts).

# Resumen y conclusiones

- Se desarrolló un abordaje a la inferencia de modelos causales basado en nociones sintácticas de correlación.
- Se implementó un algoritmo para el descubrimiento del esqueleto causal de un conjunto de datos simbólicos y se desarrollaron tests de independencia estadísticos y sintácticos.
- Se construyeron ciertos modelos funcionales con los cuales se generaron casos de prueba.
- Fueron identificadas clases de modelos cuyos esqueletos causales
  1. pueden ser inferidos empleando tests estadísticos, mas no con tests sintácticos (modelos tipo XOR);
  2. pueden ser reconstruidos mediante ambos tipos de test (modelos de concatenación); y
  3. solo pueden ser reconstruidos mediante tests sintácticos (modelos de concatenación con shifts).
- Así, los resultados sugieren que efectivamente existe un dominio de aplicabilidad específico para la inferencia causal basada en correlación sintáctica.




¡Muchas gracias!