

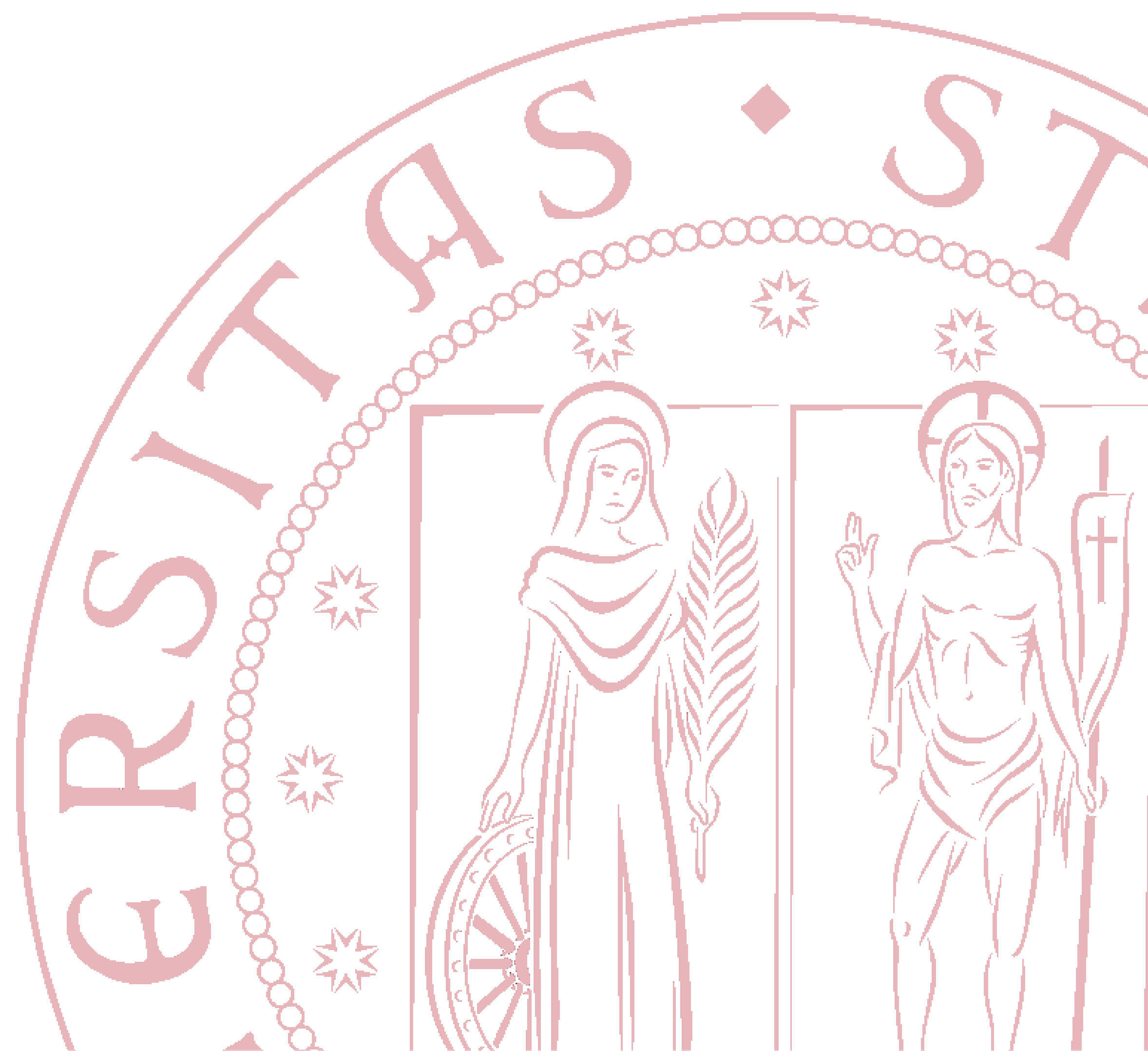
Causal Inference - Part A

Gloria Beraldo (gloria.beraldo@unipd.it)

Department of Information Engineering, University of Padova

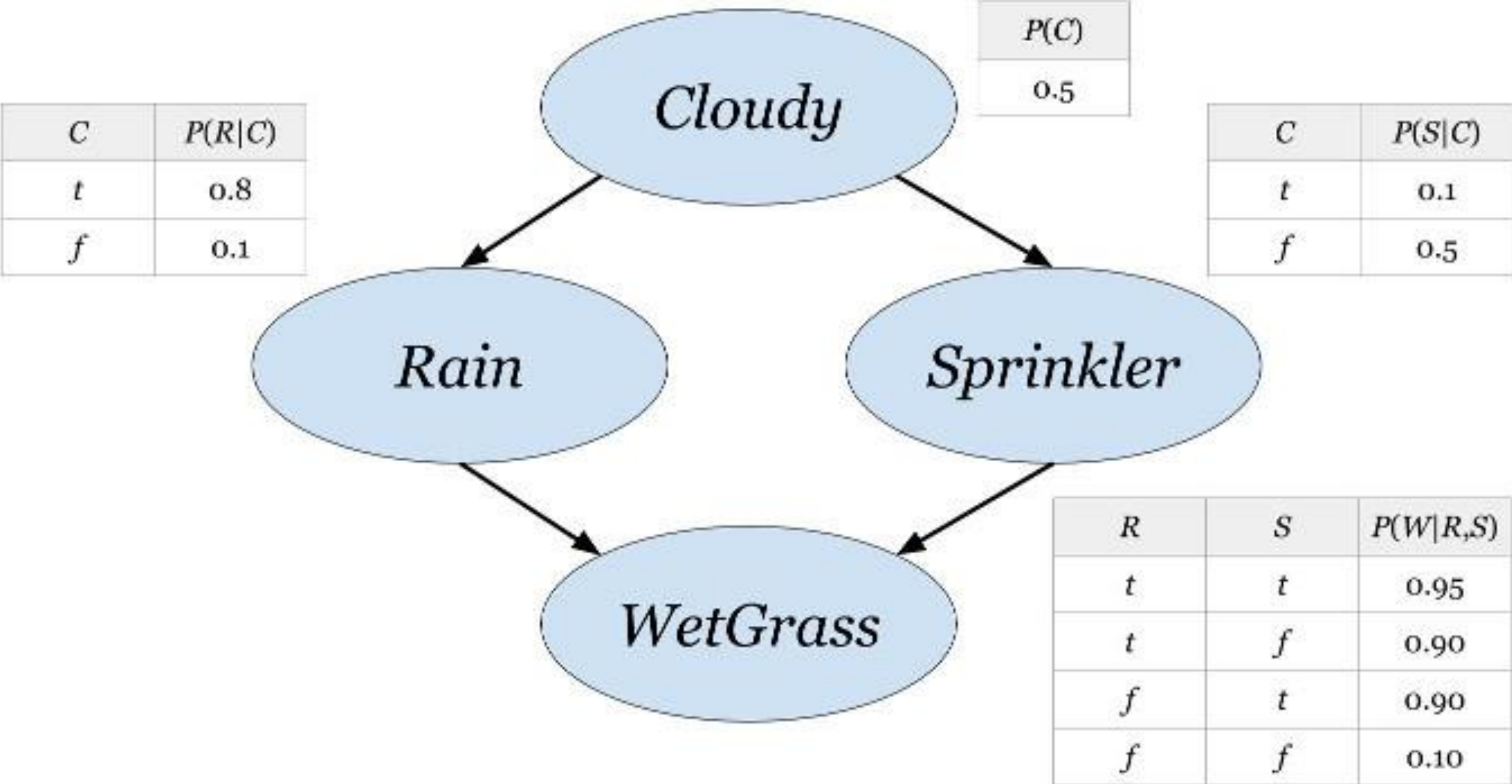
Topics:

- Causal effect of rain on wet grass: Sprinkler example
- Recap on Interventions
- Recap on adjustment formula
- pyAgrum
- Simpson's paradox
- Example Simpson's paradox via pyAgrum



Causal effect of rain on wet grass: Sprinkler example

Let's consider again our Sprinkler network, assuming this is a reliable description of the causal relationships between its four variables:



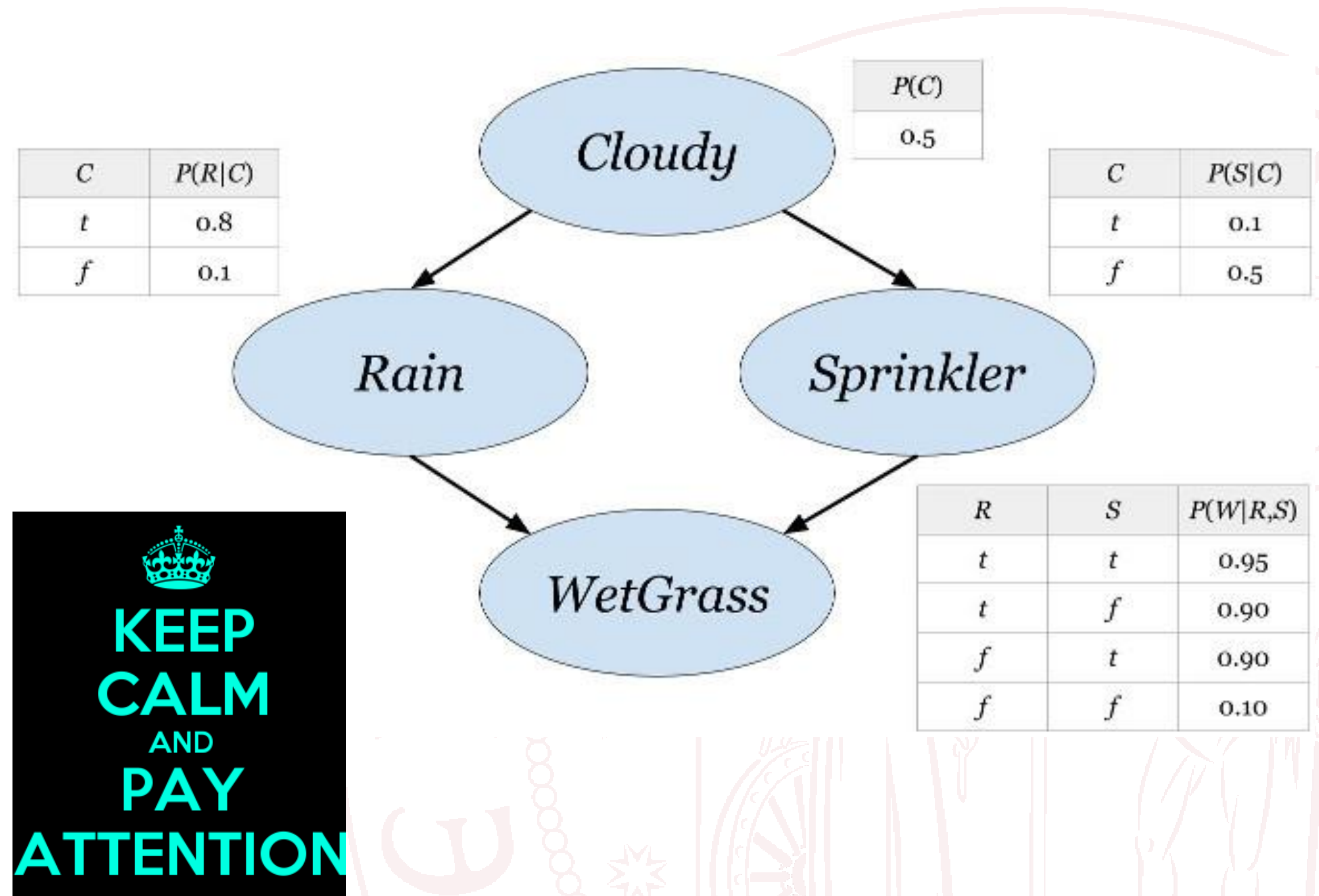
We want to estimate the **causal effects** of the rain on the "wetness" of the grass.

Causal effect of rain on wet grass: Sprinkler example

We want to estimate the **causal effects** of the rain on the "wetness" of the grass.

Note that it **wouldn't be physically possible to modify the rain** variable R .

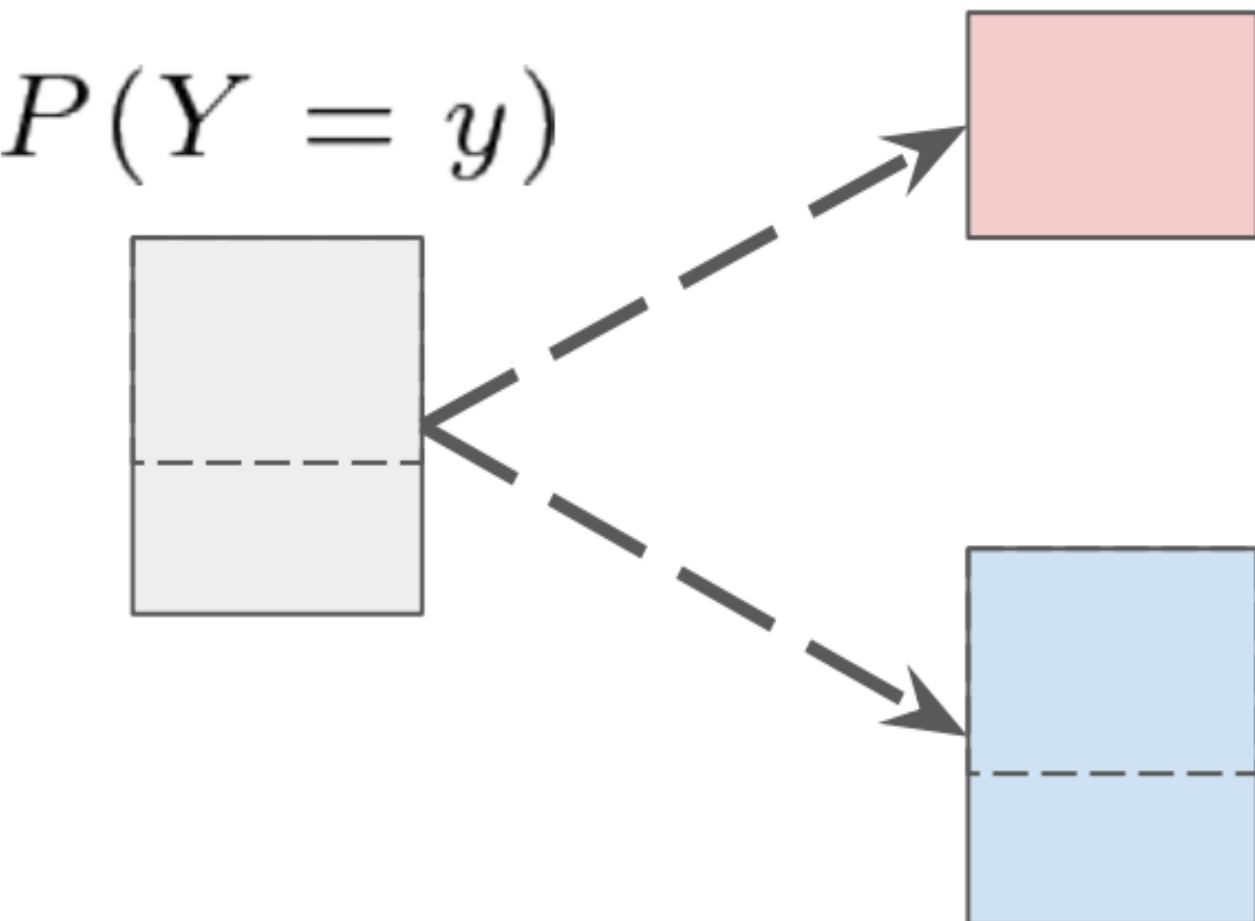
Yet, we can **use probabilities from observational data** of the weather to compute its causal effect "as if" we were able to intervene on it.



Causal effect of rain on wet grass: Sprinkler example

Interventions

- In a dataset, when we condition the outcome $Y = y$ on an **observation** $X = x$, we simply consider the subset of Y where we observe that X is equal to x
- But when we condition $Y = y$ on an **intervention** $do(X = x)$, we force the value of X for the entire set Y



$P(Y = y)$

$$P(Y = y | X = x) = \frac{P(Y = y, X = x)}{P(X = x)}$$
$$P(Y = y | do(X = x)) = ?$$



Slide 28 by prof. Bellotto

Causal effect of rain on wet grass: Sprinkler example

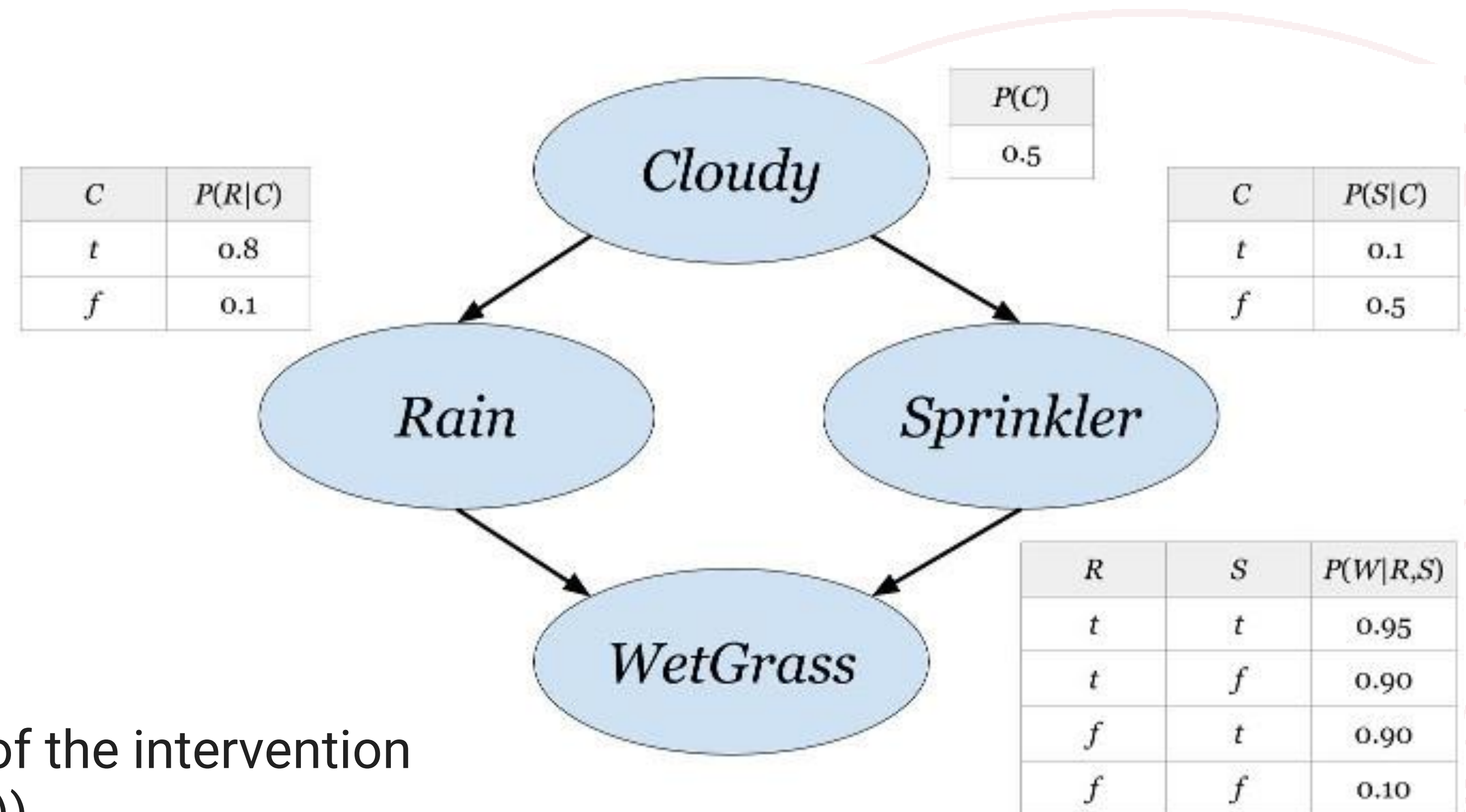
We want to estimate the **causal effects** of the rain on the "wetness" of the grass.

Note that it **wouldn't be physically possible to modify the rain** variable R .

Yet, we can **use probabilities from observational data** of the weather to compute its causal effect "as if" we were able to intervene on it.

To this end, we can compute the effect of the intervention $P(G=\text{true}|\text{do}(R=\text{true}))$, or simply $P(g|\text{do}(r))$,

by using the **adjustment formula** for the only parent of R , which is C



Causal effect of rain on wet grass: Sprinkler example

Adjustment formula

$$P(y|do(x)) = P_m(y|x)$$

from definition of intervention

$$= \sum_z P_m(y|x, z)P_m(z|x)$$

from Law of Total Probability

$$= \sum_z P_m(y|x, z)P_m(z)$$

from independence of X and Z

$$= \sum_z P(y|x, z)P(z)$$

from previous slide's equalities

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

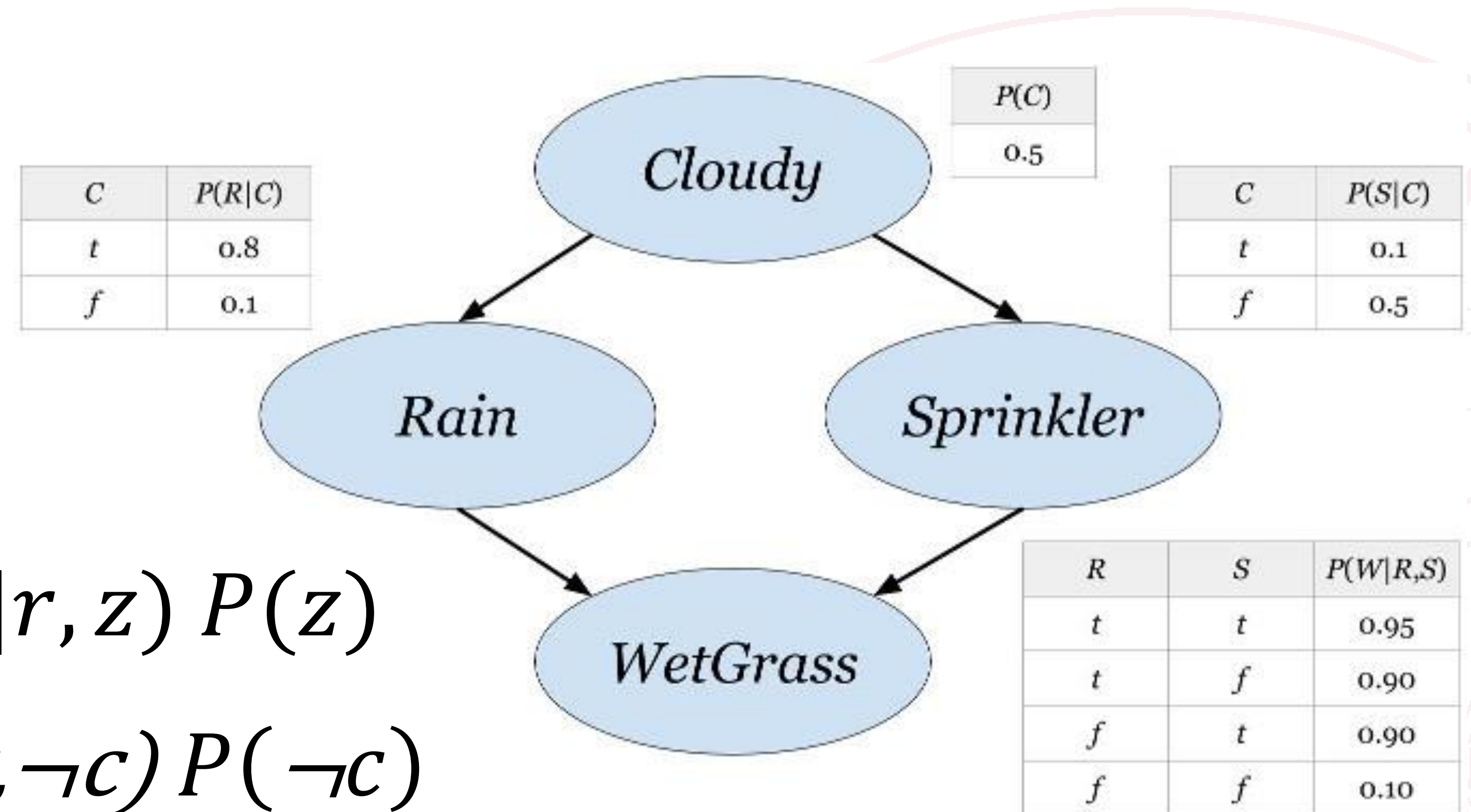
$$P(y|do(x)) = \sum_{z \in \Lambda} P(y|x, z)P(z)$$

where Λ is the set of parents of X



Causal effect of rain on wet grass: Sprinkler example

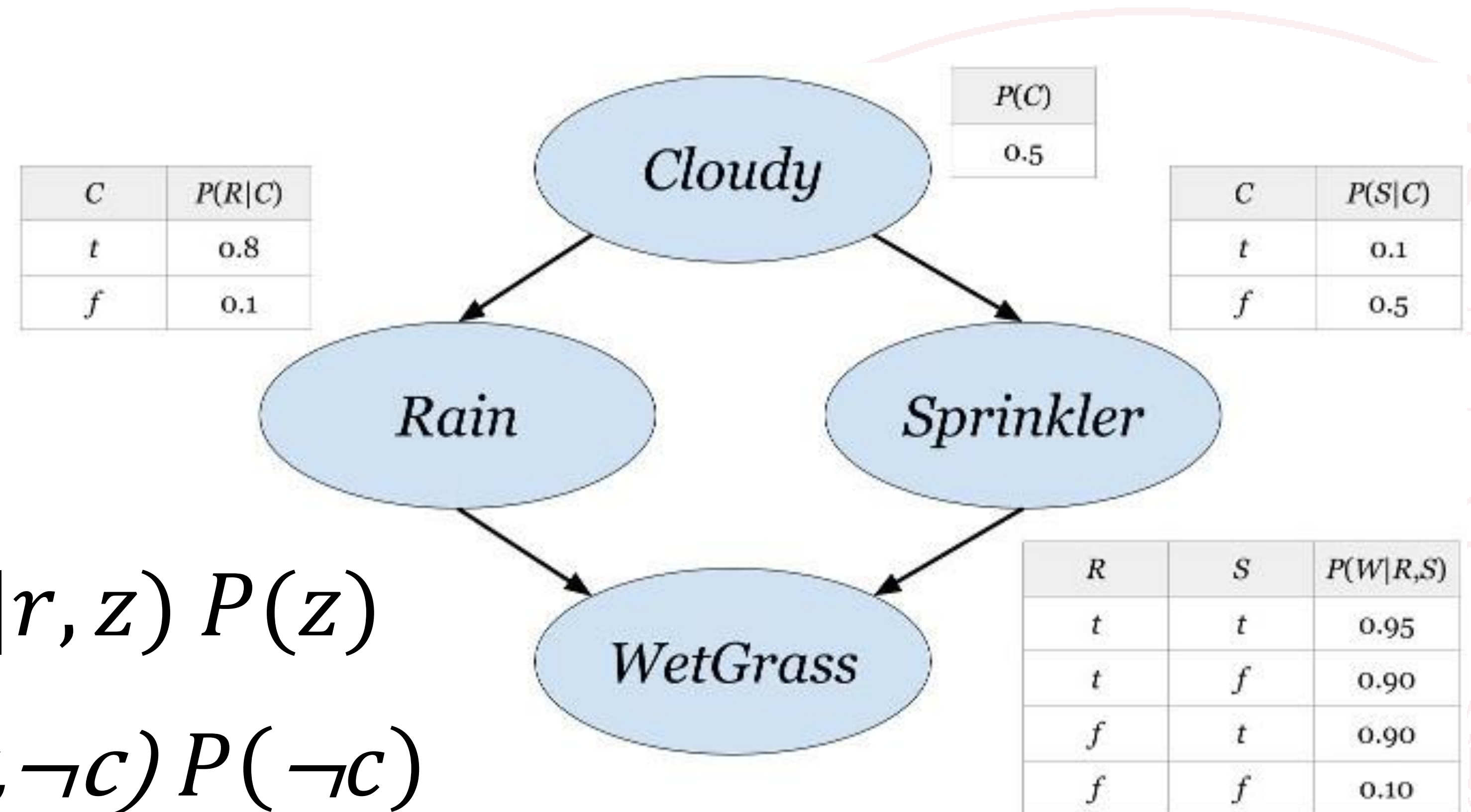
We want to estimate the **causal effects** of the rain on the "wetness" of the grass.



$$\begin{aligned} P(g/do(r)) &= \sum_{z \in C} P(g|r, z) P(z) \\ &= P(g/r, c) P(c) + P(g/r, \neg c) P(\neg c) \end{aligned}$$

Causal effect of rain on wet grass: Sprinkler example

We want to estimate the **causal effects** of the rain on the "wetness" of the grass.

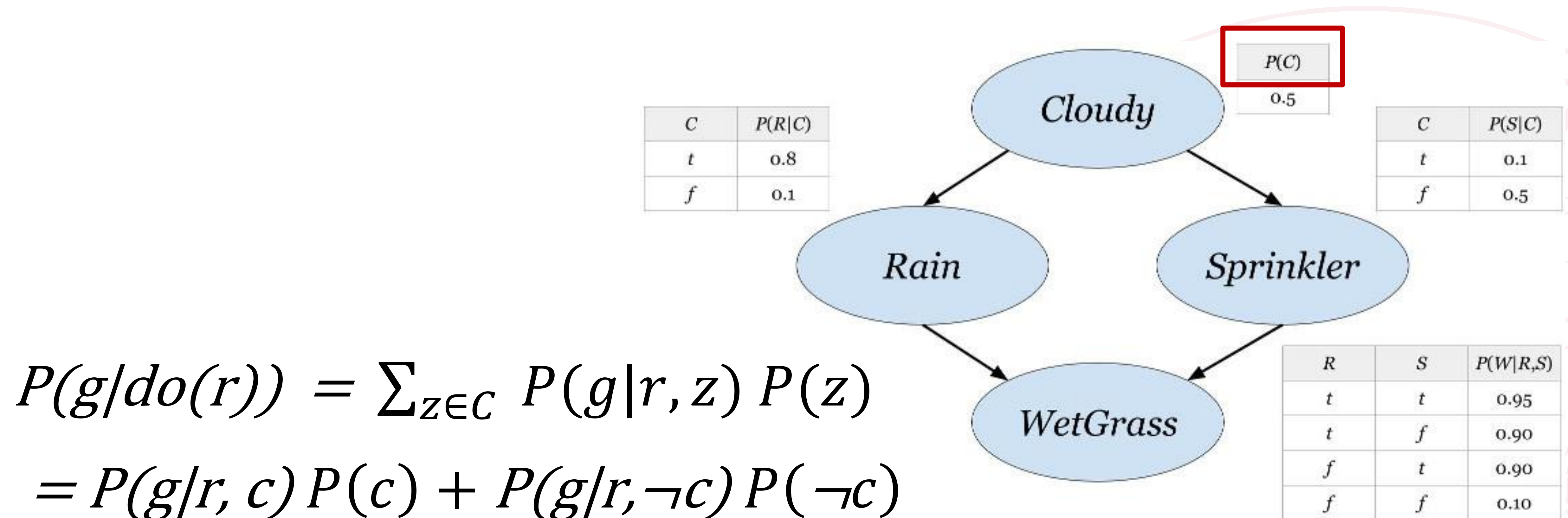


$$\begin{aligned} P(g/do(r)) &= \sum_{z \in C} P(g|r, z) P(z) \\ &= P(g/r, c) P(c) + P(g/r, \neg c) P(\neg c) \end{aligned}$$

The probability distribution $P(C) = \langle P(c), P(\neg c) \rangle$ is already given by the network.

Causal effect of rain on wet grass: Sprinkler example

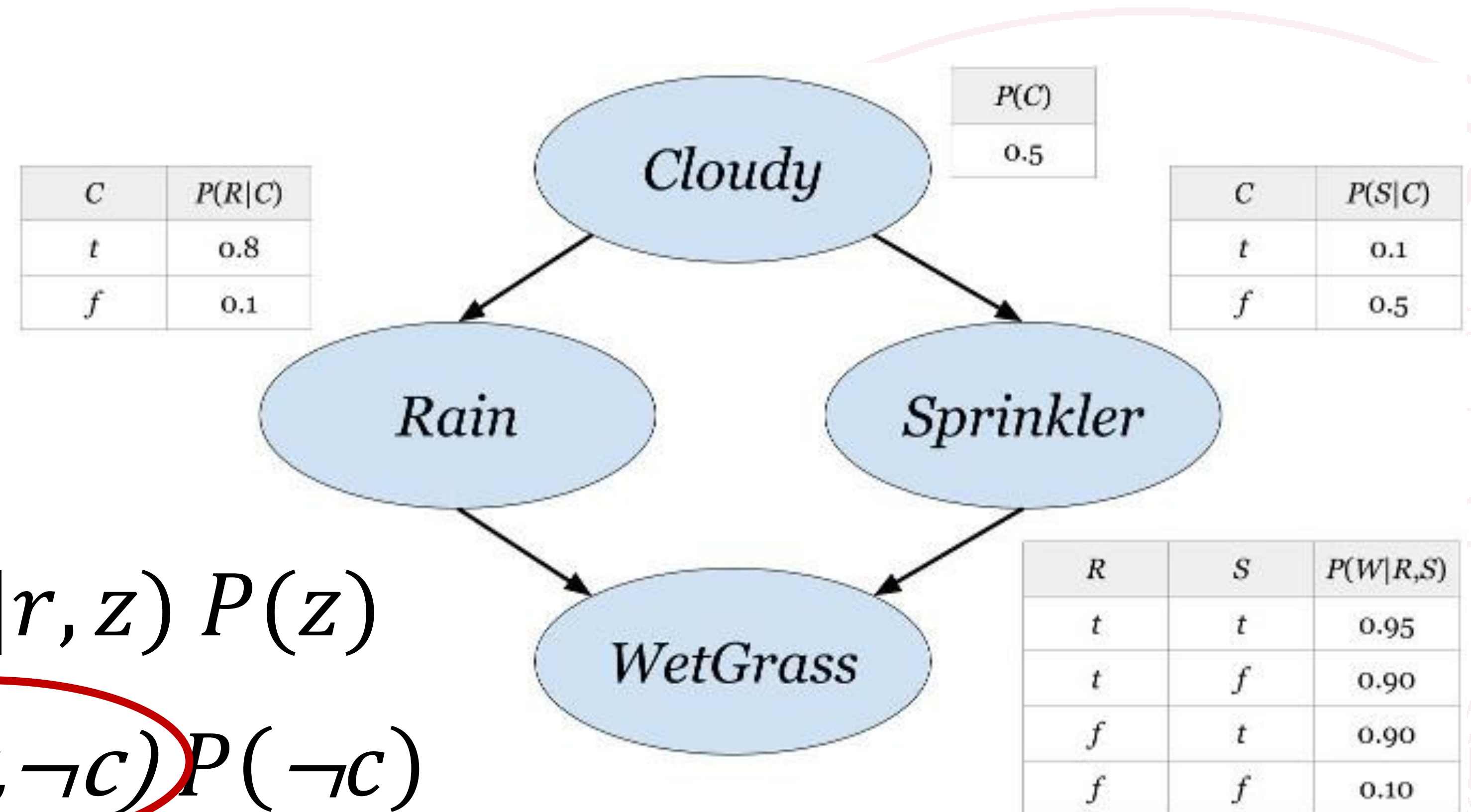
We want to estimate the **causal effects** of the rain on the "wetness" of the grass.



The probability distribution $\mathbf{P}(C) = \langle P(c), P(\neg c) \rangle$ is already given by the network.

Causal effect of rain on wet grass: Sprinkler example

We want to estimate the **causal effects** of the rain on the "wetness" of the grass.



$$\begin{aligned} P(g/do(r)) &= \sum_{z \in C} P(g|r, z) P(z) \\ &= P(g/r, c) P(c) + P(g/r, \neg c) P(\neg c) \end{aligned}$$

We need to compute $\mathbf{P}(G|r, c) = \langle P(g|r, c), P(\neg g|r, c) \rangle$

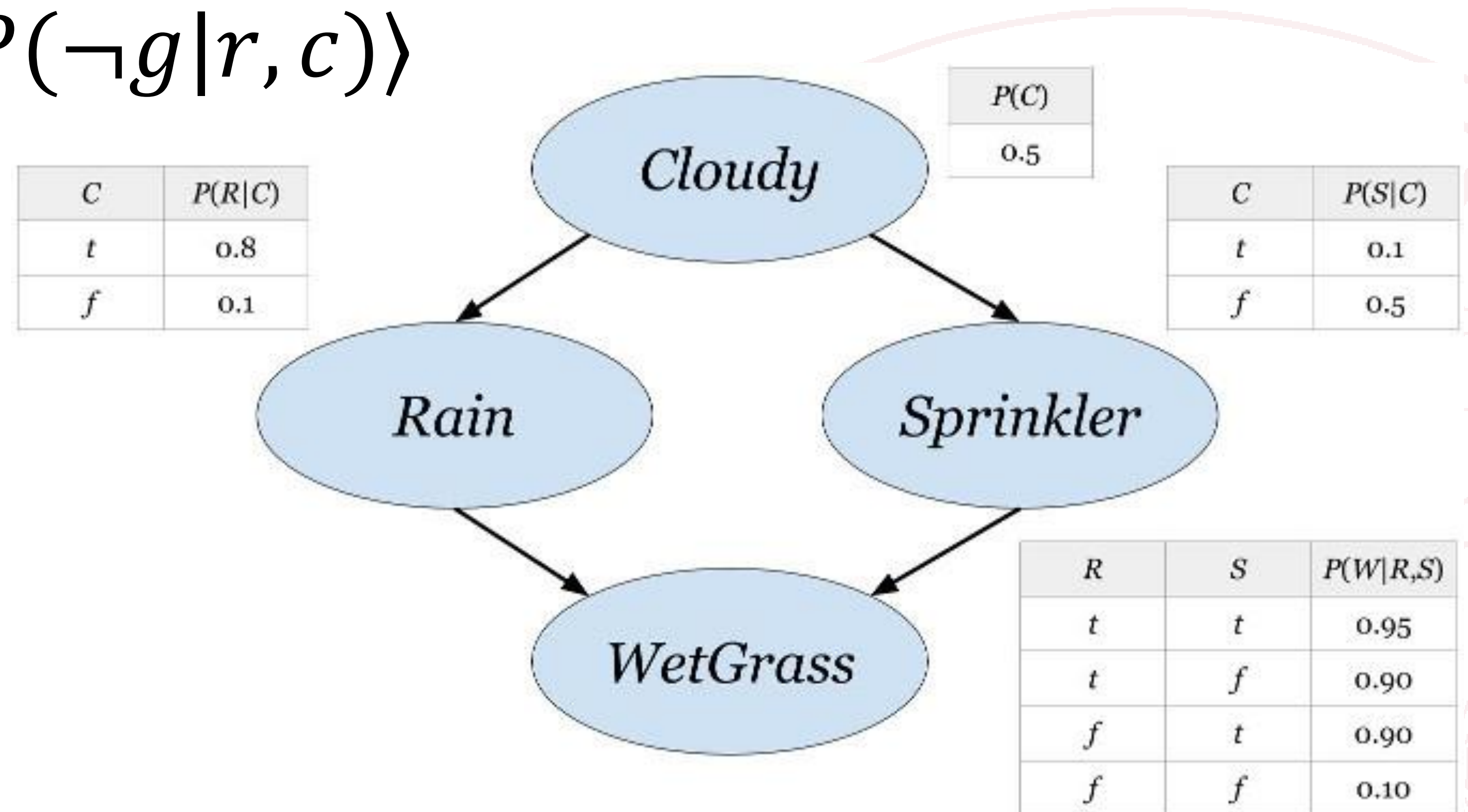
Causal effect of rain on wet grass: Sprinkler example

The conditional distribution

$$\mathbf{P}(G|r, c) = \langle P(g|r, c), P(\neg g|r, c) \rangle$$

can be computed as follows:

$$\begin{aligned}\mathbf{P}(G|r, c) &= \frac{P(G, r, c)}{P(r, c)} \\ &= \alpha P(G, r, c)\end{aligned}$$

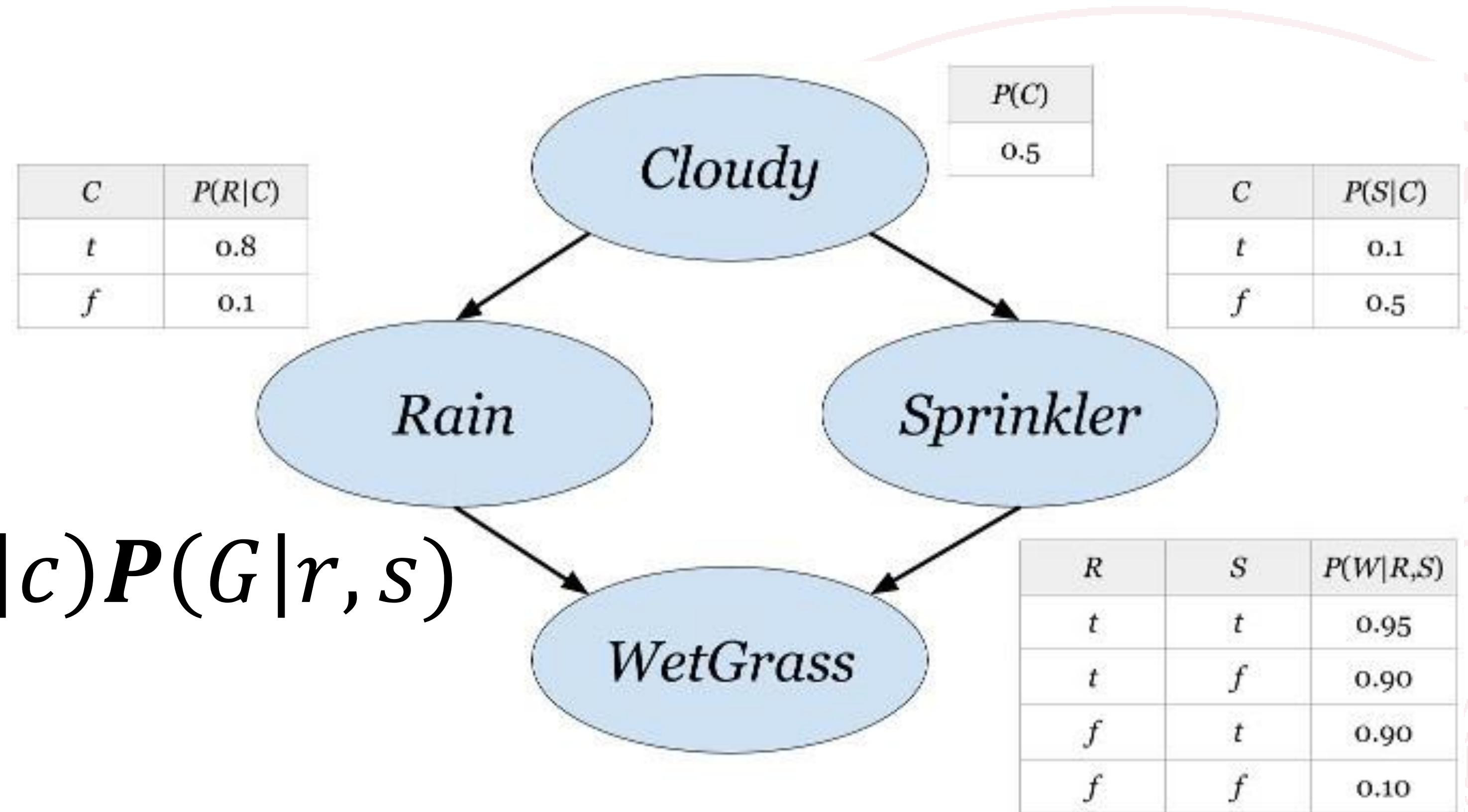


Causal effect of rain on wet grass: Sprinkler example

$$= \alpha P(G, r, c)$$

$$= \alpha \sum_s P(G, r, c, s)$$

$$= \alpha \sum_s P(c)P(r|c)P(s|c)P(G|r, s)$$



As we did in Lab 5

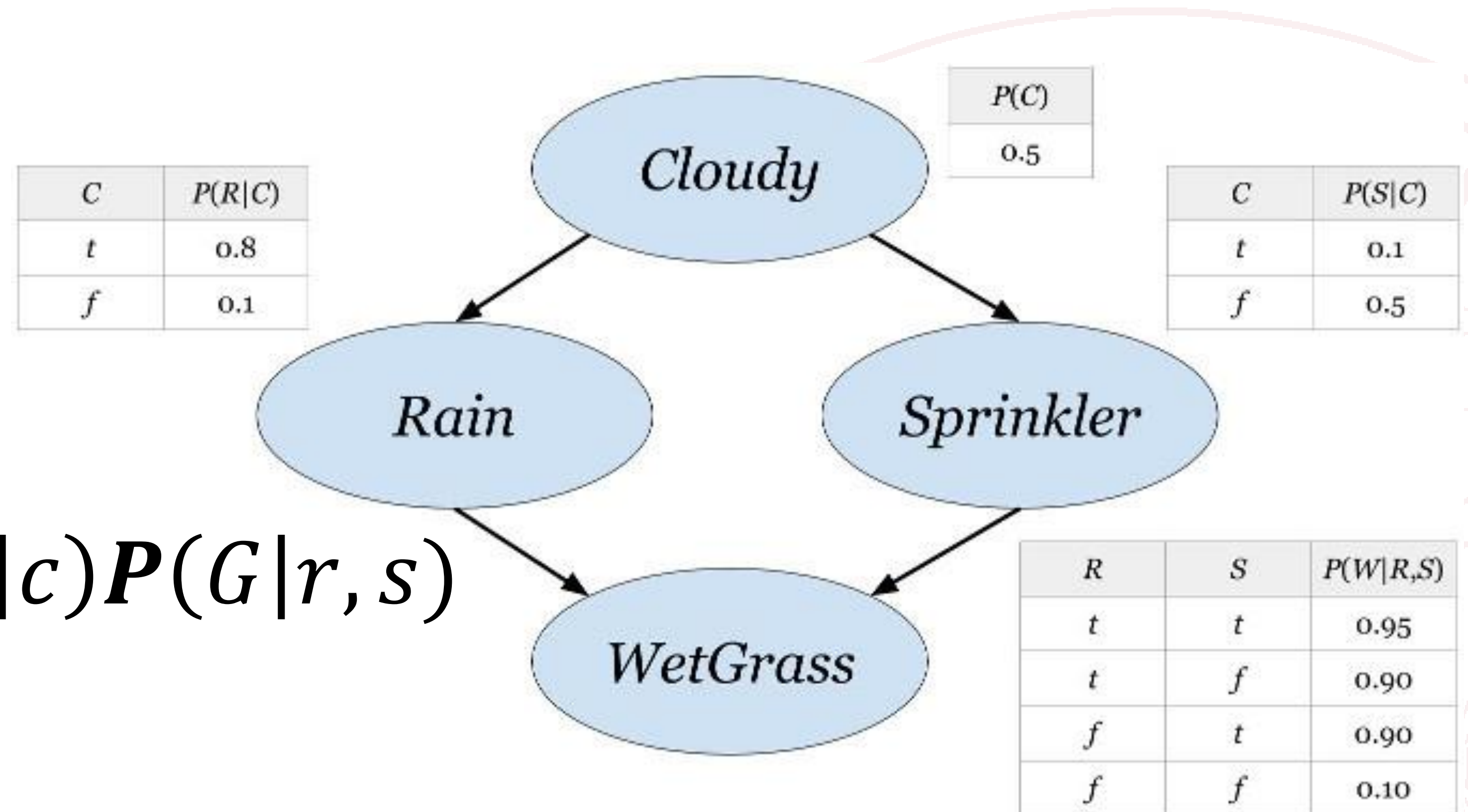
Causal effect of rain on wet grass: Sprinkler example

$$= \alpha P(G, r, c)$$

$$= \alpha \sum_s P(G, r, c, s)$$

$$= \alpha \sum_s P(c)P(r|c)P(s|c)P(G|r, s)$$

$$= \alpha P(c) \boxed{P(r|c)} \sum_s P(s|c)P(G|r, s)$$



Causal effect of rain on wet grass: Sprinkler example

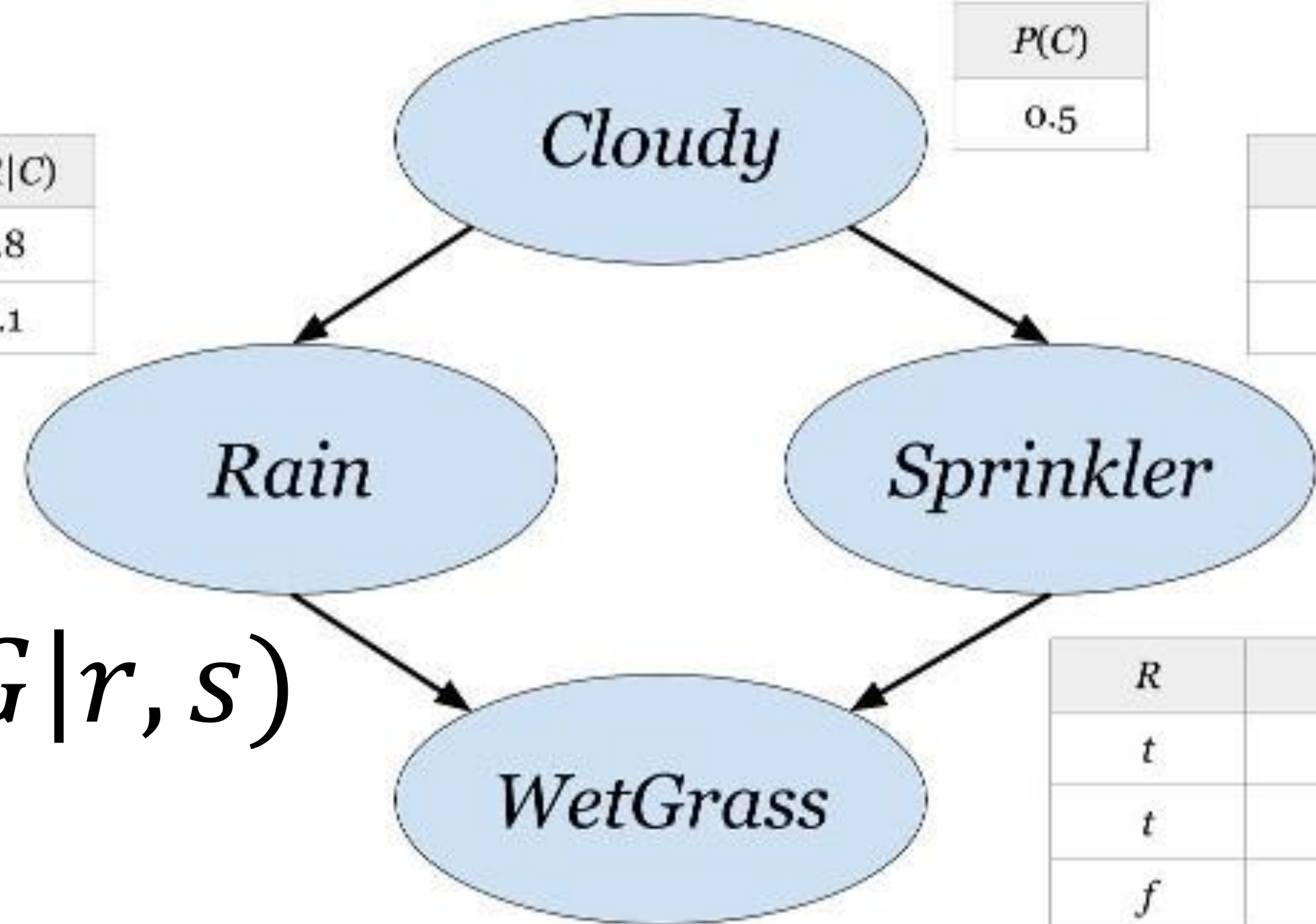
$$= \alpha P(G, r, c)$$

$$= \alpha \sum_s P(G, r, c, s)$$

$$= \alpha \sum_s P(c)P(r|c)P(s|c)P(G|r, s)$$

$$= \underbrace{\alpha P(c)P(r|c)}_{\alpha'} \sum_s P(s|c)P(G|r, s)$$

C	$P(R C)$
t	0.8
f	0.1



$P(C)$
0.5

C	$P(S C)$
t	0.1
f	0.5

R	S	$P(W R,S)$
t	t	0.95
t	f	0.90
f	t	0.90
f	f	0.10

α' is a new normalization factor

Causal effect of rain on wet grass: Sprinkler example

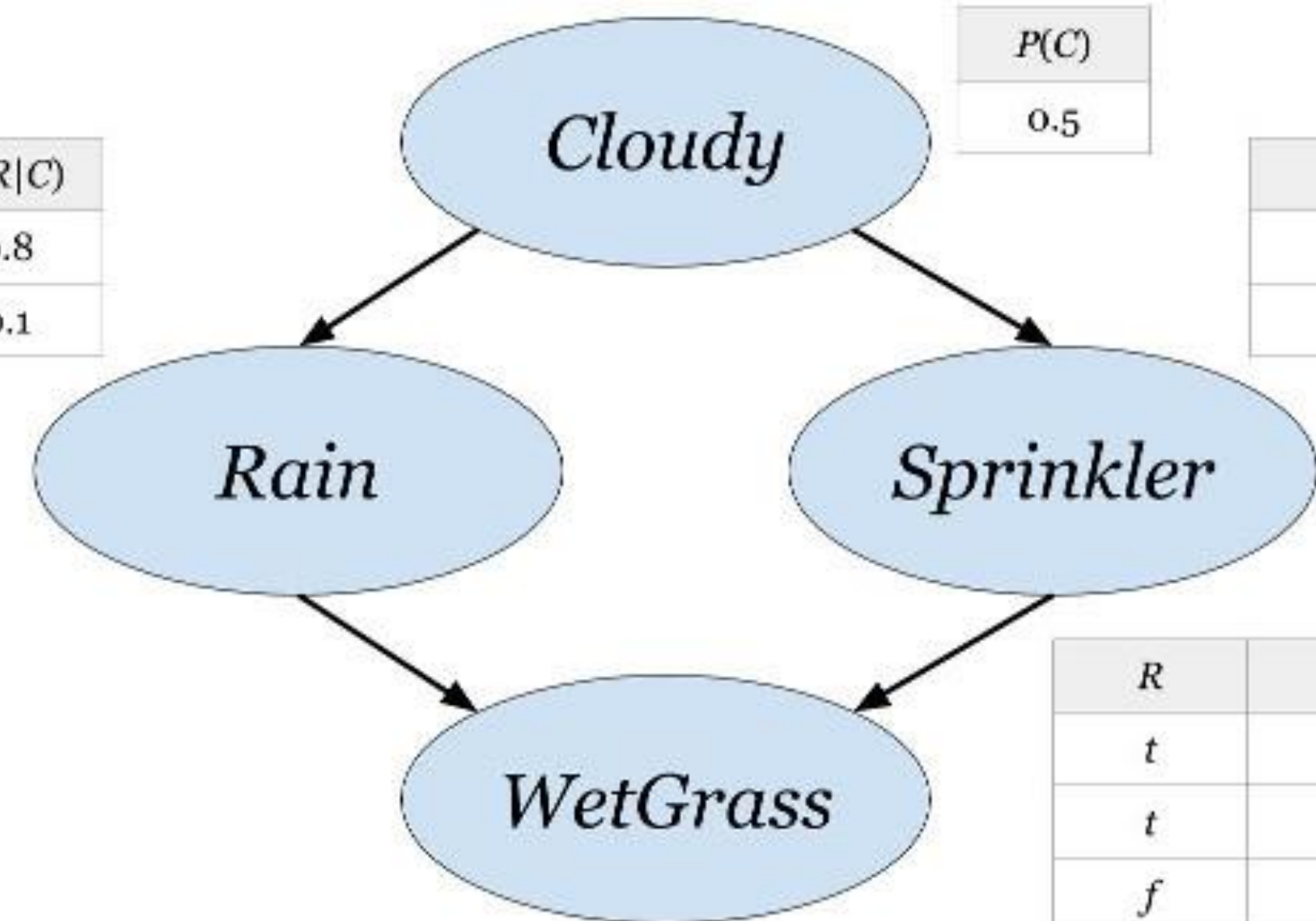
$$= \alpha' \sum_s P(s|c) P(G|r, s)$$

$$= \alpha' [P(s|c)P(G|r,s) + P(\neg s|c)P(G|r,\neg s)]$$

Substituting the values from the network's CPTs, we get the following:



<i>C</i>	$P(R C)$
<i>t</i>	0.8
<i>f</i>	0.1



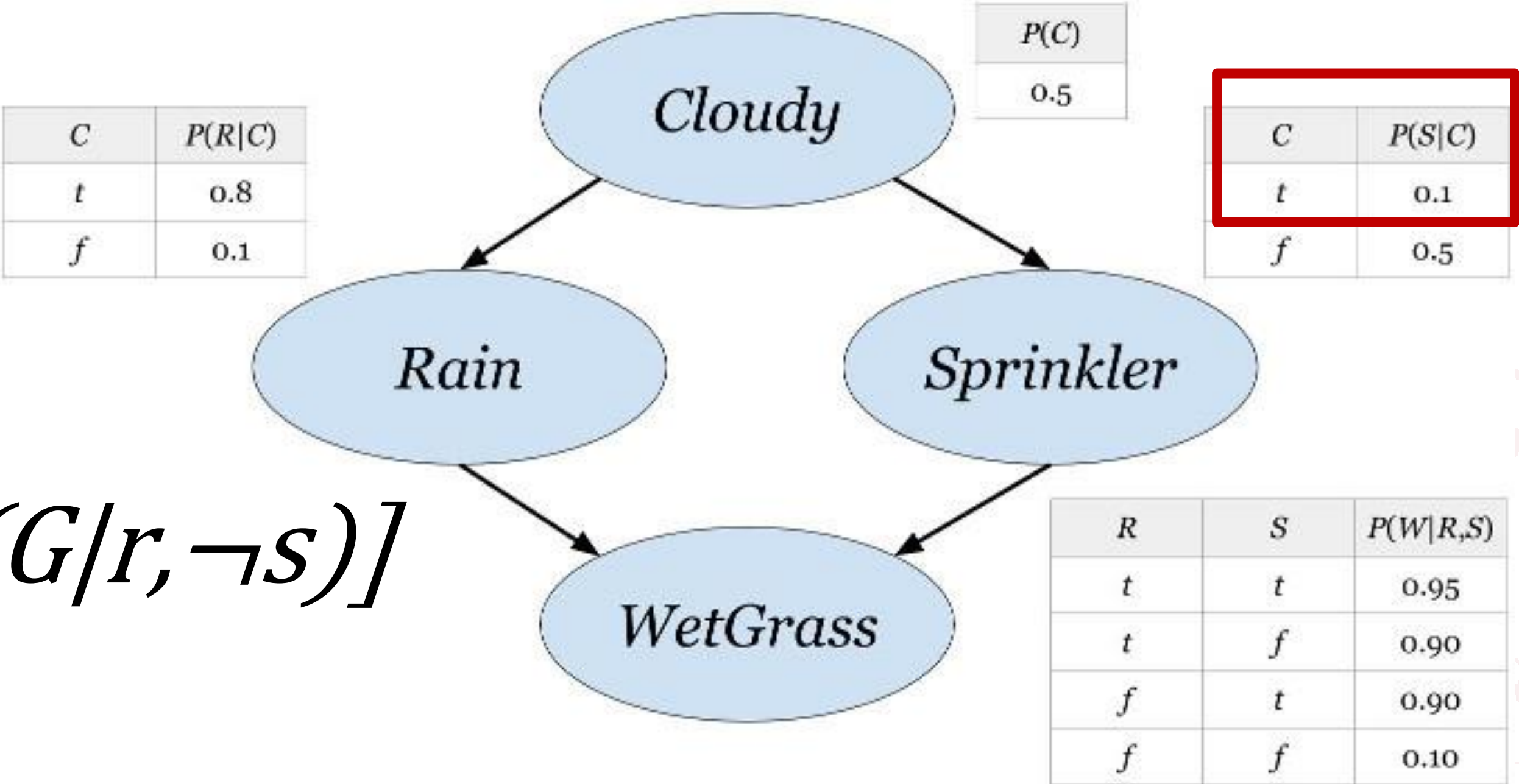
$P(C)$
0.5

<i>C</i>	$P(S C)$
<i>t</i>	0.1
<i>f</i>	0.5

<i>R</i>	<i>S</i>	$P(W R,S)$
<i>t</i>	<i>t</i>	0.95
<i>t</i>	<i>f</i>	0.90
<i>f</i>	<i>t</i>	0.90
<i>f</i>	<i>f</i>	0.10

Causal effect of rain on wet grass: Sprinkler example

Substituting the values from the network's CPTs, we get the following:



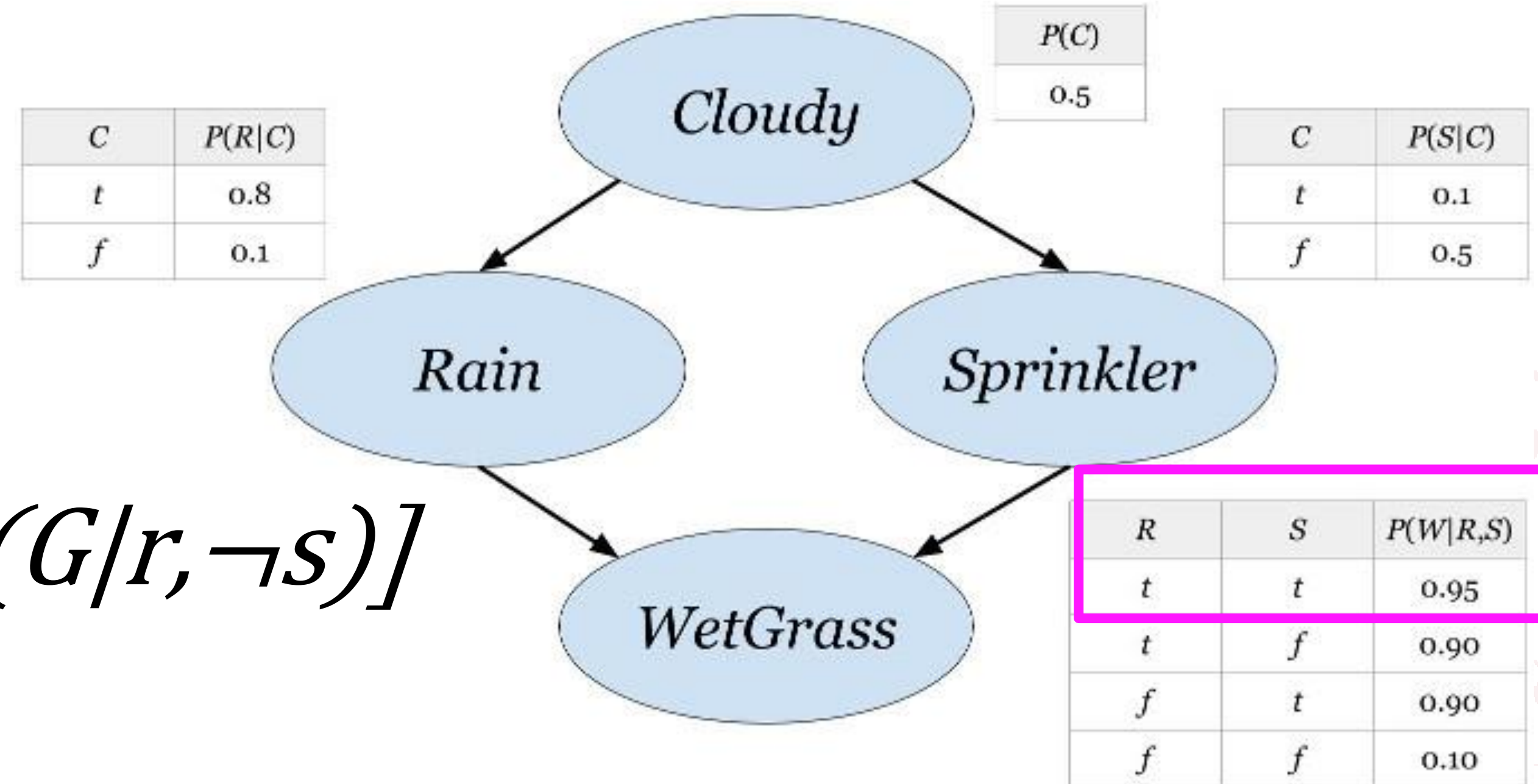
$$= \alpha' [P(s/c)P(G/r,s)+P(\neg s/c)P(G/r,\neg s)]$$

$$\begin{aligned} P(G|r,c) &= \\ \alpha' [0.1 \times \langle 0.95,0.05 \rangle + 0.9 \times \langle 0.90,0.10 \rangle] \\ &= \langle 0.905,0.095 \rangle \end{aligned}$$



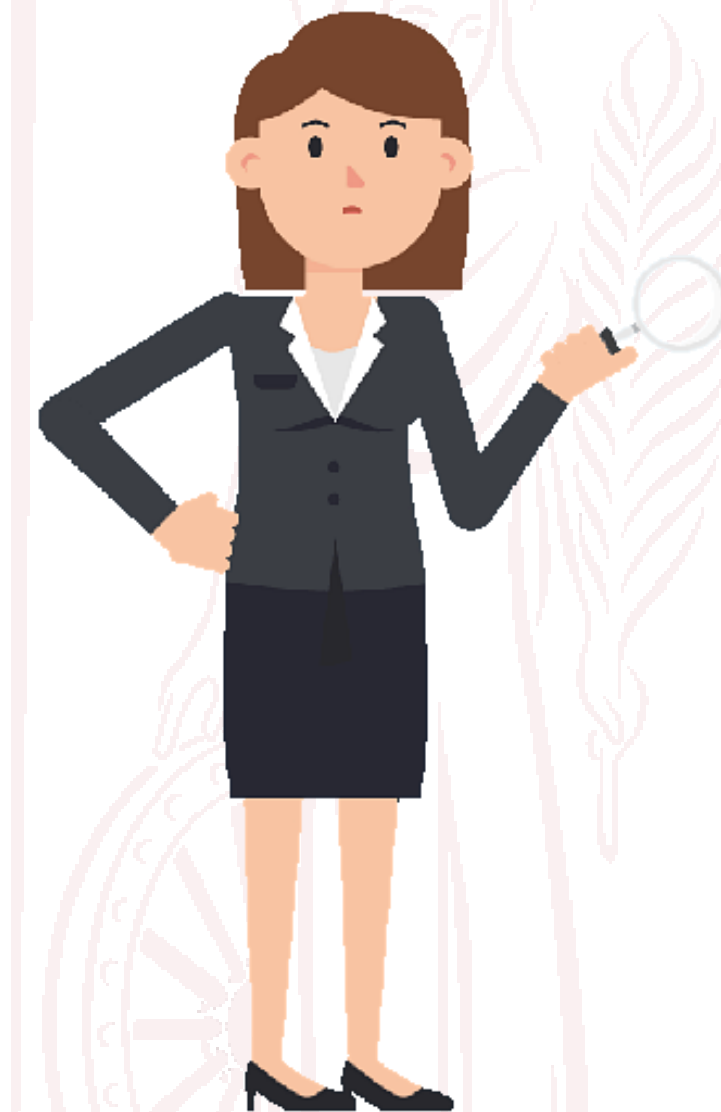
Causal effect of rain on wet grass: Sprinkler example

Substituting the values from the network's CPTs, we get the following:



$$= \alpha' [P(s/c)P(G/r,s) + P(\neg s/c)P(G/r,\neg s)]$$

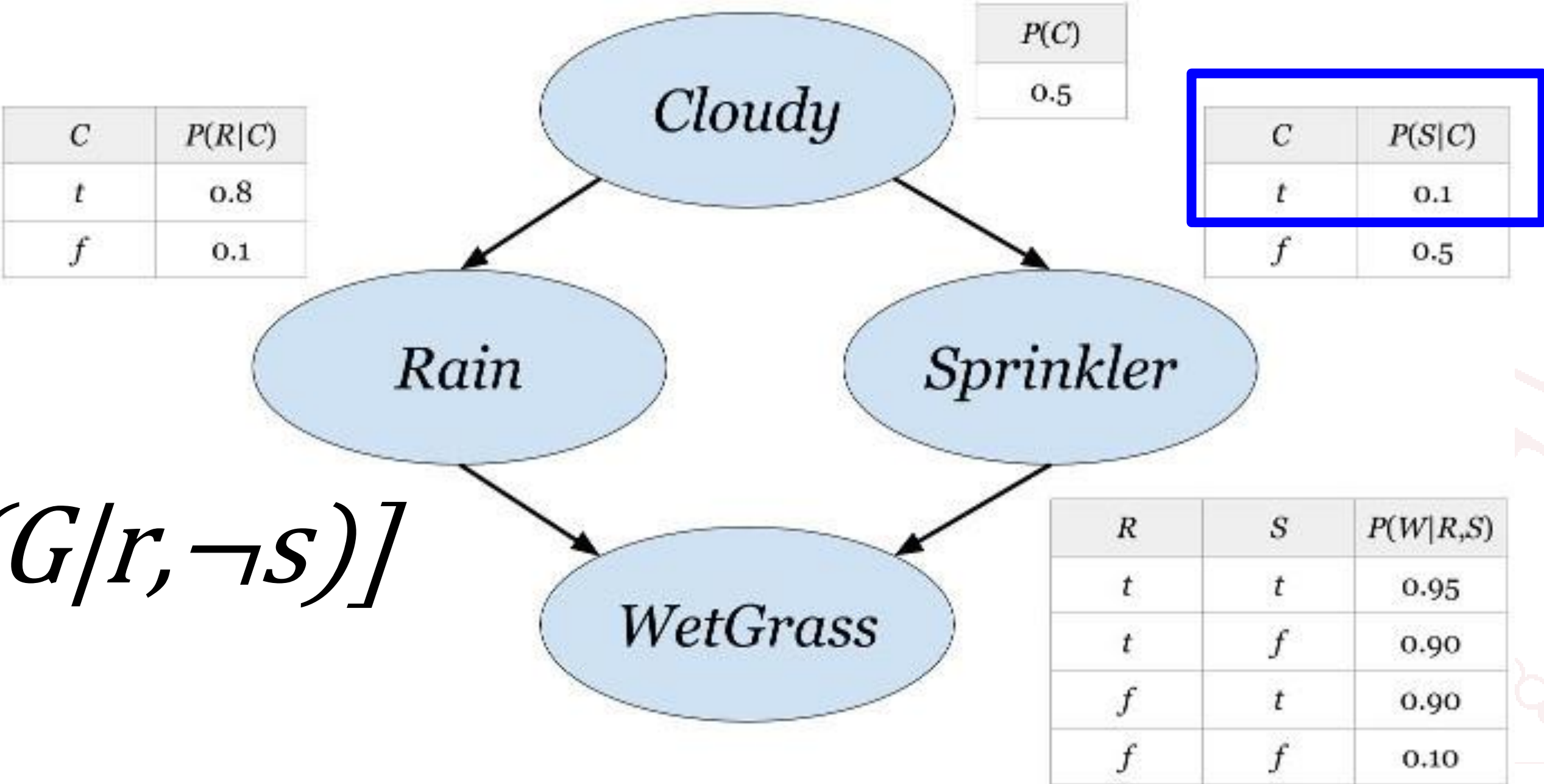
$$\begin{aligned} P(G|r,c) &= \\ \alpha' [0.1 \times \langle 0.95, 0.05 \rangle + 0.9 \times \langle 0.90, 0.10 \rangle] \\ &= \langle 0.905, 0.095 \rangle \end{aligned}$$



Causal effect of rain on wet grass: Sprinkler example

Substituting the values from the network's CPTs, we get the following:

$$= \alpha' [P(s/c)P(G/r,s) + \boxed{P(\neg s/c)P(G/r,\neg s)}]$$



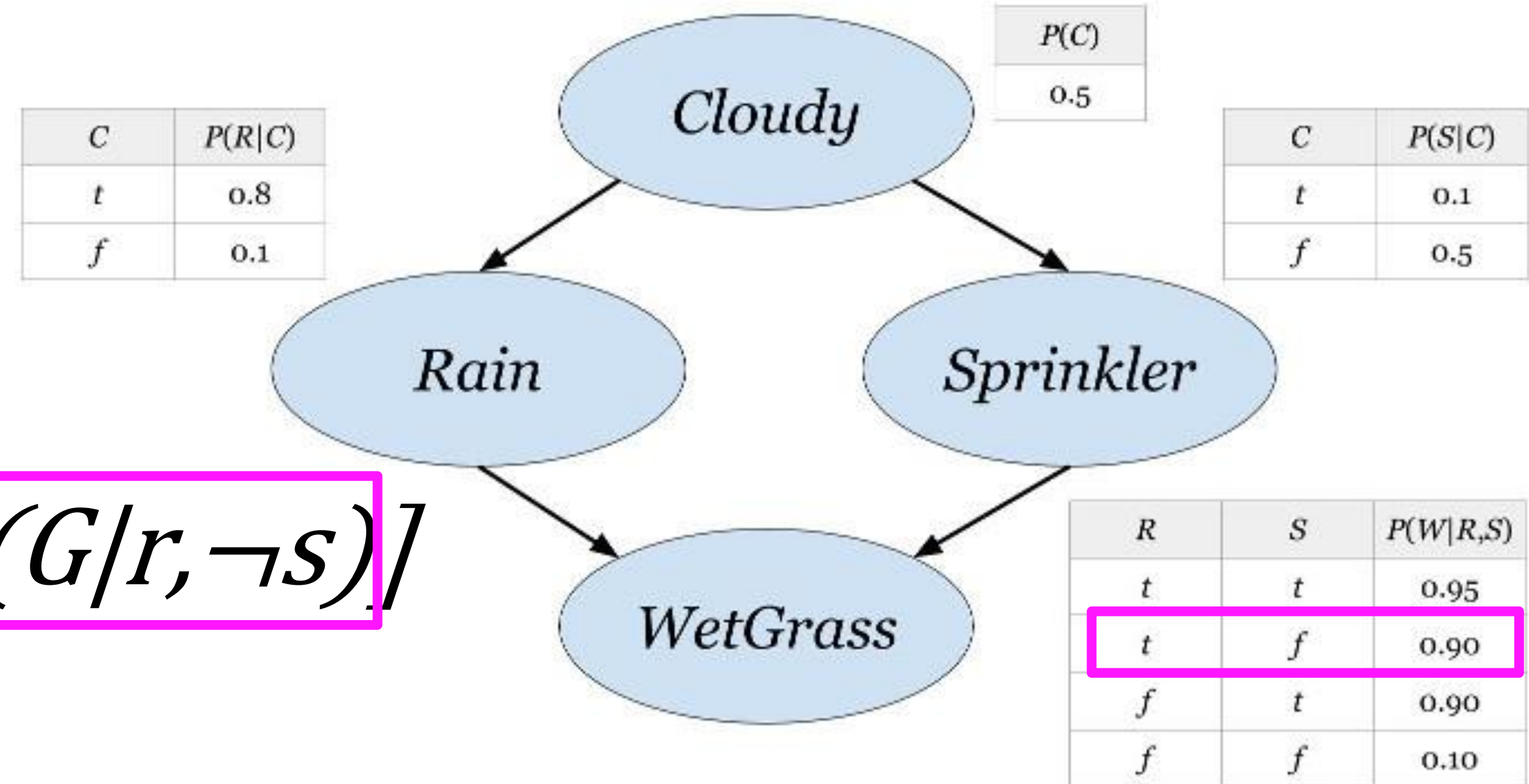
$$\begin{aligned} P(G|r,c) &= \\ \alpha' [0.1 \times \langle 0.95, 0.05 \rangle + 0.9 \times \langle 0.90, 0.10 \rangle] \\ &= \langle 0.905, 0.095 \rangle \end{aligned}$$



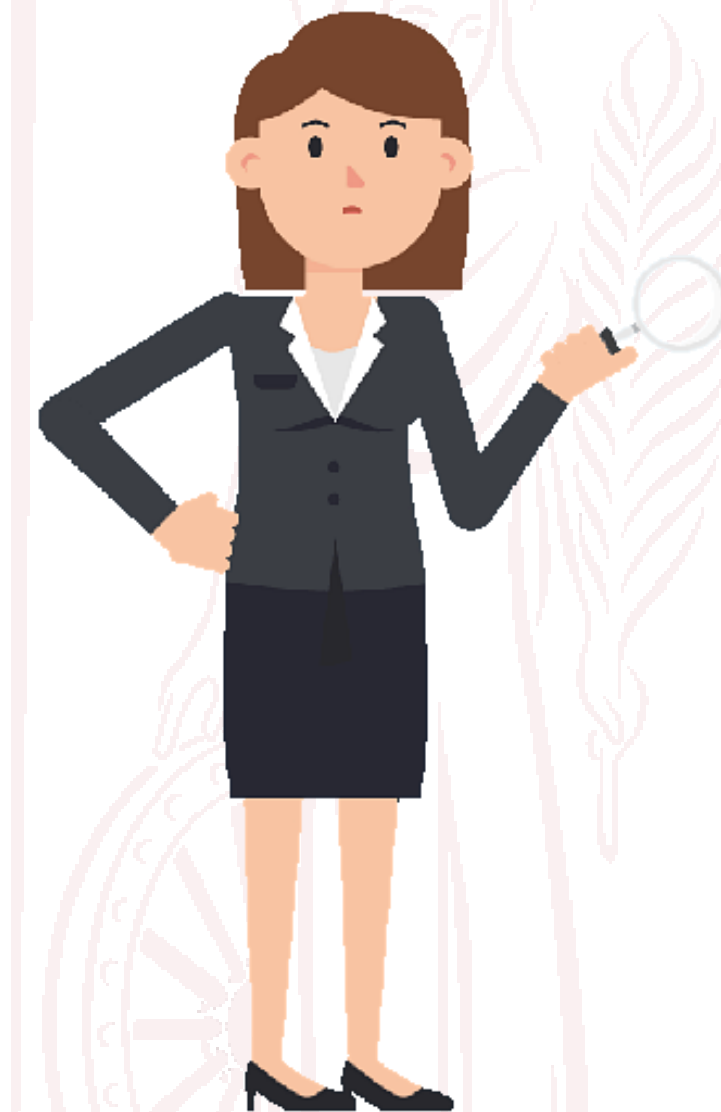
Causal effect of rain on wet grass: Sprinkler example

Substituting the values from the network's CPTs, we get the following:

$$= \alpha' [P(s/c)P(G/r,s) + P(\neg s/c)P(G/r,\neg s)]$$

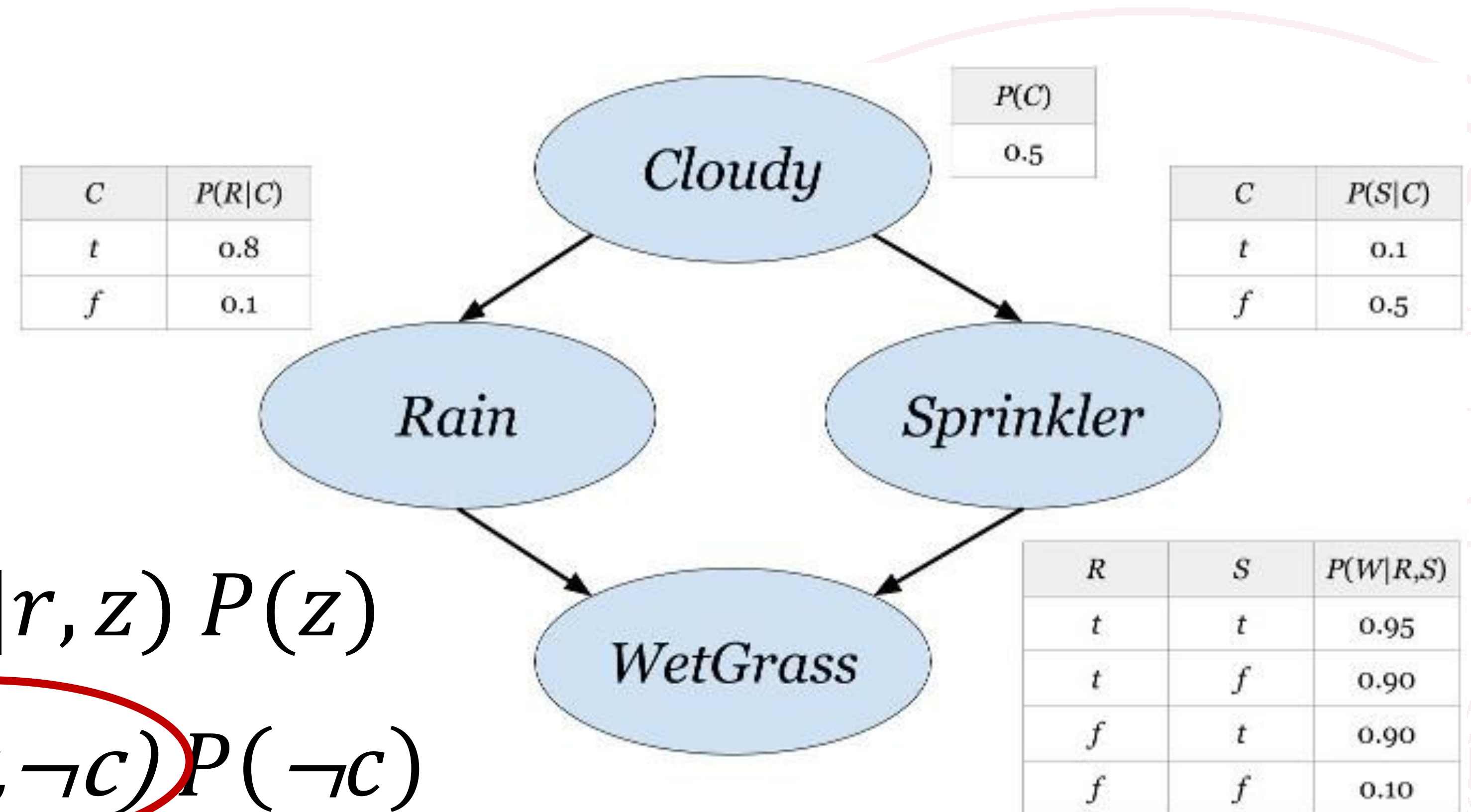


$$\begin{aligned} P(G|r,c) &= \\ \alpha' [0.1 \times \langle 0.95, 0.05 \rangle + 0.9 \times \langle 0.90, 0.10 \rangle] \\ &= \langle 0.905, 0.095 \rangle \end{aligned}$$



Causal effect of rain on wet grass: Sprinkler example

We want to estimate the **causal effects** of the rain on the "wetness" of the grass.



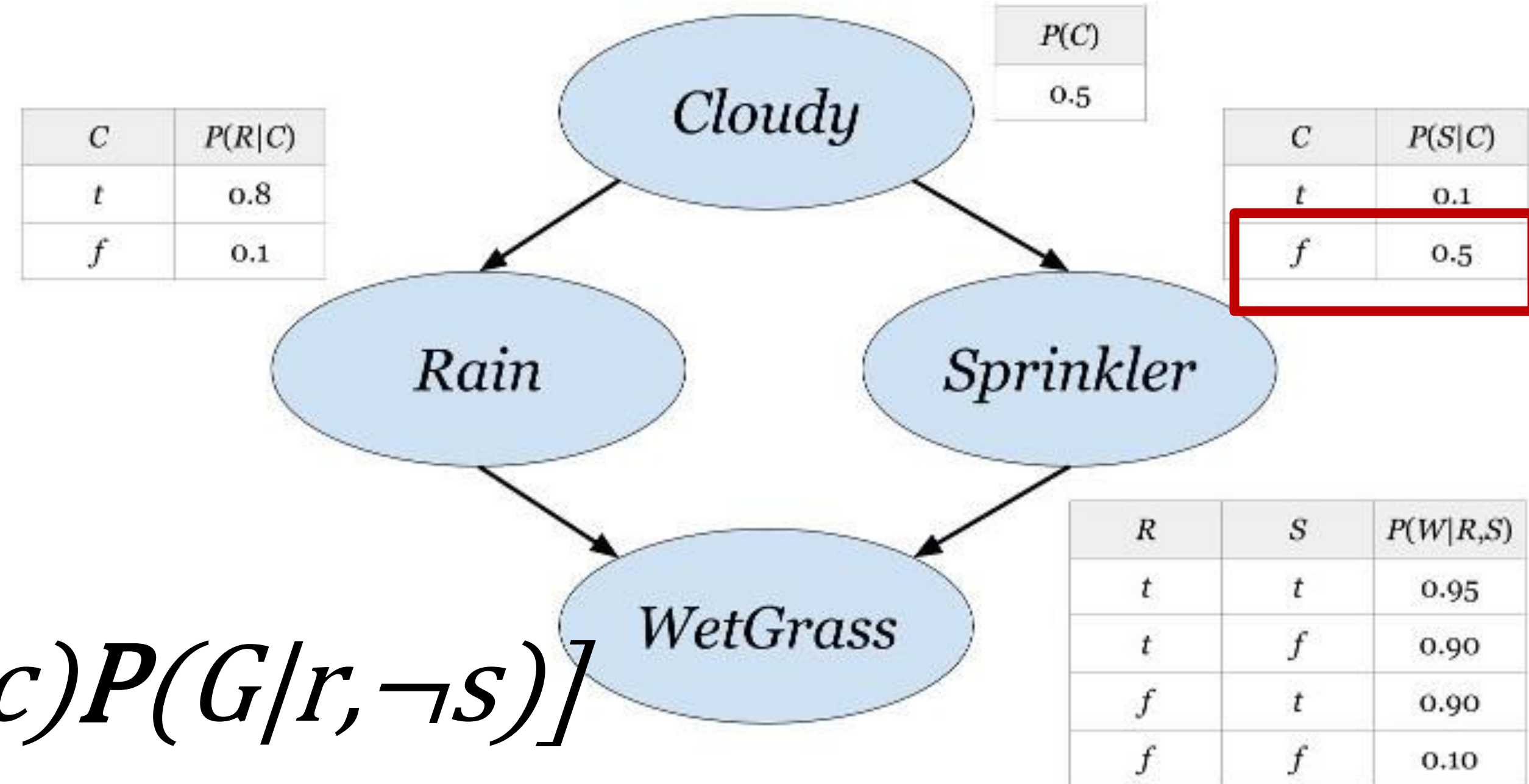
$$\begin{aligned} P(g/do(r)) &= \sum_{z \in C} P(g|r, z) P(z) \\ &= P(g/r, c) P(c) + P(g/r, \neg c) P(\neg c) \end{aligned}$$



We need to compute $P(G|r, \neg c) = \langle P(g|r, \neg c), P(\neg g|r, \neg c) \rangle$

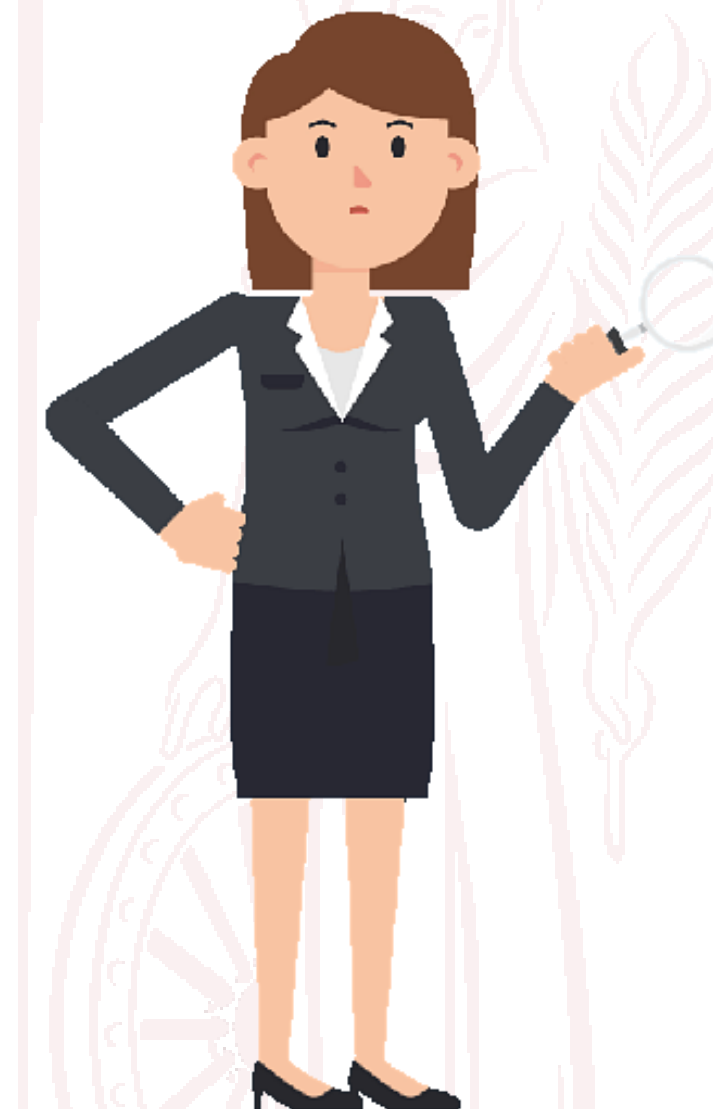
Causal effect of rain on wet grass: Sprinkler example

If we do the same for the condition $\neg c$, we obtain the following distribution



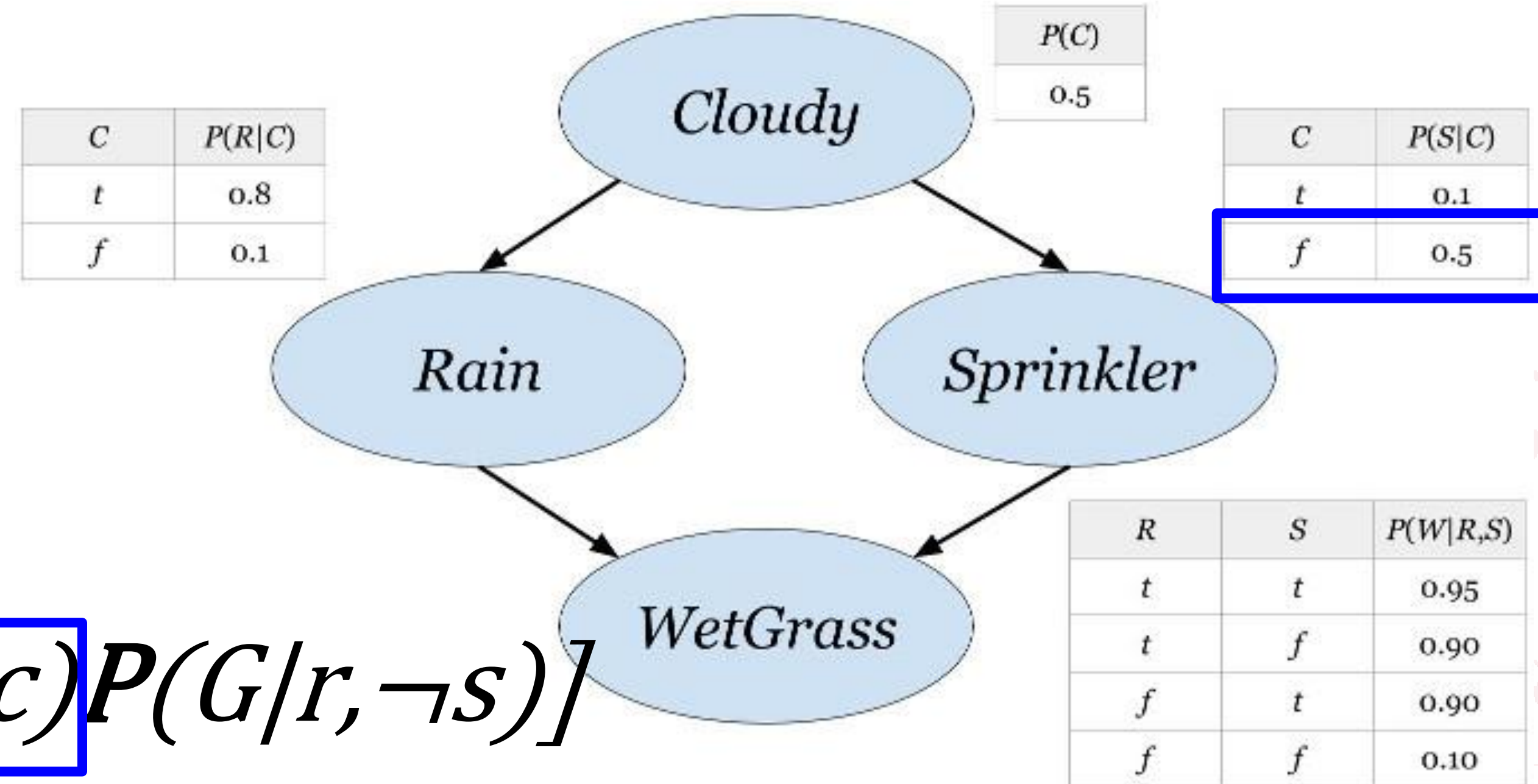
$$P(G|r, \neg c) = \alpha' [P(s/\neg c)P(G/r,s) + P(\neg s/\neg c)P(G/r,\neg s)]$$

$$P(G|r, \neg c) = \alpha' [0.5 \times \langle 0.95, 0.05 \rangle + 0.5 \times \langle 0.90, 0.10 \rangle] = \langle 0.925, 0.075 \rangle$$



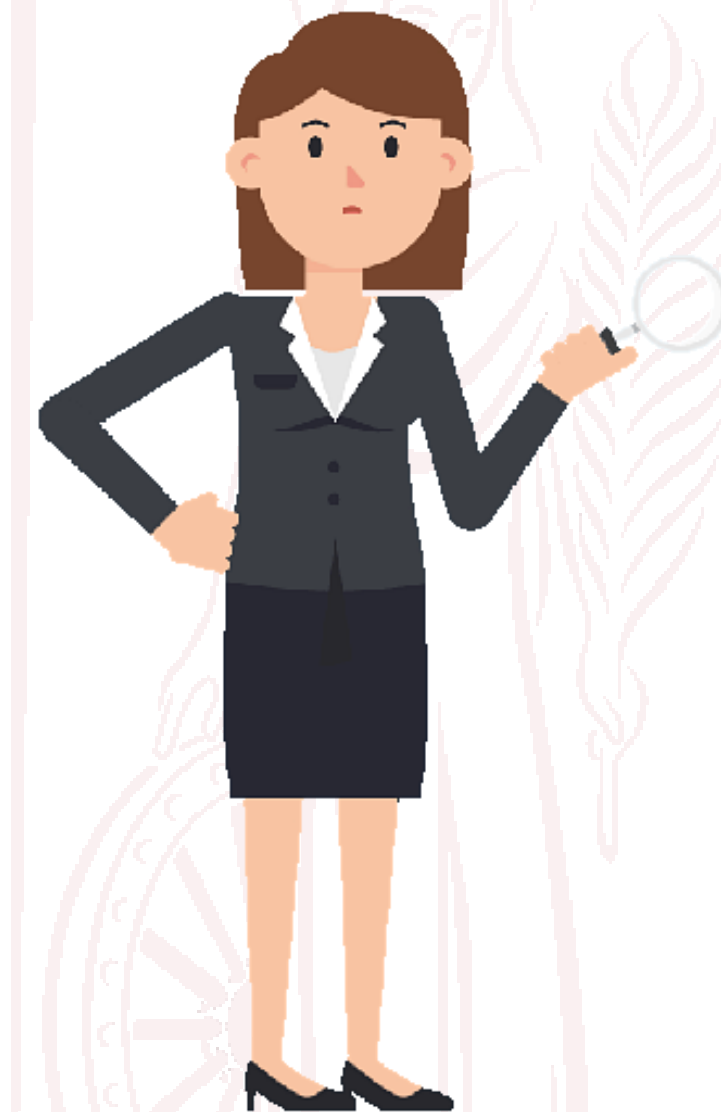
Causal effect of rain on wet grass: Sprinkler example

If we do the same for the condition $\neg c$, we obtain the following distribution



$$P(G|r, \neg c) = \alpha' [P(s/\neg c)P(G/r,s) + P(\neg s/\neg c)P(G/r,\neg s)]$$

$$P(G|r, \neg c) = \alpha' [0.5 \times \langle 0.95, 0.05 \rangle + 0.5 \times \langle 0.90, 0.10 \rangle] = \langle 0.925, 0.075 \rangle$$



Causal effect of rain on wet grass: Sprinkler example

Finally, we use the calculated values,

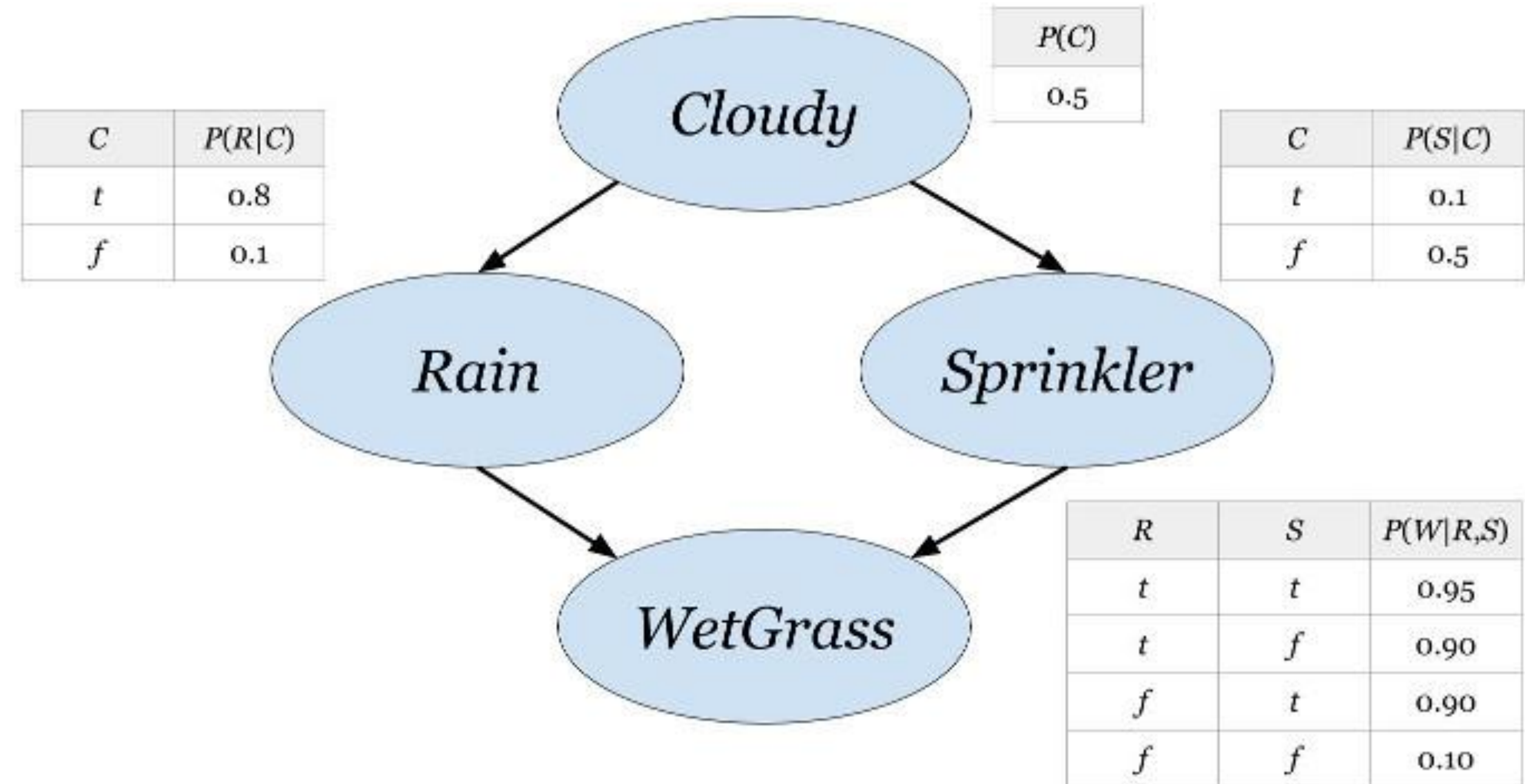
$$P(g|r, c) = 0.905$$

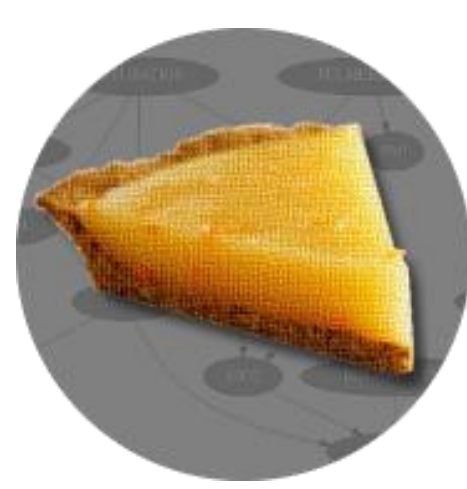
$$P(g|r, \neg c) = 0.925$$

in the previous adjustment formula
and obtain the following:

$$\begin{aligned} P(g/do(r)) &= \sum_{z \in \mathcal{C}} P(g|r, z) P(z) \\ &= P(g/r, c) P(c) + P(g/r, \neg c) P(\neg c) \\ &= 0.905 \times 0.5 + 0.925 \times 0.5 = 0.915 \end{aligned}$$

which is our causal effect of the intervention $R=\text{true}$ on the wetness $G=\text{true}$.





pyAgrum is a scientific C++ and Python library dedicated to Bayesian networks (BN) and other Probabilistic Graphical Models.

Based on the C++ aGrUM library, it provides a high-level interface to the C++ part of aGrUM allowing to create, manage and perform efficient computations with Bayesian networks and others probabilistic graphical models:

Markov random fields (MRF),
influence diagrams (ID) and LIMIDs,
credal networks (CN),
dynamic BN (dBN),
probabilistic relational models (PRM).

<https://pyagrum.readthedocs.io/en/latest/index.html>



Simpson's paradox

Simpson's paradox is a phenomenon in probability and statistics in which a trend appears in several groups of data but disappears or reverses when the groups are combined.

- A new medicine was offered to 700 patients: 350 of them chose to take it, while 350 did not.

	Medicine	No medicine
Men	81 out of 87 recovered (93%)	234 out of 270 recovered (87%)
Women	192 out of 263 recovered (73%)	55 out of 80 recovered (69%)
Combined data	273 out of 350 recovered (78%)	289 out of 350 recovered (83%)

- The medicine worked for the two subgroups, men and women, but not for the population as a whole. How is that possible?

Simpson's paradox via pyAgrum

Let's apply pyAgrum to the Simpson's paradox with this example

```
!pip install pyAgrum
from IPython.display import display, Math, Latex

import pyAgrum as gum
import pyAgrum.lib.notebook as gnb
import pyAgrum.causal as cs1
import pyAgrum.causal.notebook as cs1nb
```

Looking in indexes: <https://pypi.org/simple>, <https://us-python.pkg.dev/colab-wheels/public/simple/>

Collecting pyAgrum

Downloading pyAgrum-1.7.1-cp39-cp39-manylinux2014_x86_64.whl (5.6 MB)

5.6/5.6 MB 13.1 MB/s eta 0:00:00

Requirement already satisfied: numpy in /usr/local/lib/python3.9/dist-packages (from pyAgrum) (1.22.4)
Requirement already satisfied: pydot in /usr/local/lib/python3.9/dist-packages (from pyAgrum) (1.4.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.9/dist-packages (from pyAgrum) (3.7.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.9/dist-packages (from matplotlib->pyAgrum) (1.4.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib->pyAgrum) (23.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.9/dist-packages (from matplotlib->pyAgrum) (1.0.7)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.9/dist-packages (from matplotlib->pyAgrum) (0.11.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.9/dist-packages (from matplotlib->pyAgrum) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.9/dist-packages (from matplotlib->pyAgrum) (2.8.2)
Requirement already satisfied: importlib-resources>=3.2.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib->pyAgrum) (5.12.0)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib->pyAgrum) (8.4.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib->pyAgrum) (4.39.3)
Requirement already satisfied: zipp>=3.1.0 in /usr/local/lib/python3.9/dist-packages (from importlib-resources>=3.2.0->matplotlib->pyAgrum) (3.15.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from python-dateutil>=2.7->matplotlib->pyAgrum) (1.16.0)
Installing collected packages: pyAgrum
Successfully installed pyAgrum-1.7.1

<https://pyagrums.readthedocs.io/en/latest/index.html>



Simpson's paradox via pyAgrum

In a statistical study about a drug, we try to evaluate the latter's efficiency among a population of men and women.

Let's note: - Drug : drug taking - Patient : cured patient - Gender : patient's gender

The model from the observed data is as follow :

```
m1 = gum.fastBN("Gender{F|M}->Drug{Without|With}->Patient{Sick|Healed}<-Gender")

m1.cpt("Gender")[:] = [0.5, 0.5]
m1.cpt("Drug")[:] = [[0.25, 0.75], #Gender=F
                    [0.75, 0.25]] #Gender=M

m1.cpt("Patient")[{ 'Drug': 'Without', 'Gender': 'F' }] = [0.2, 0.8] #No Drug, Male -> healed in 0.8 of cases
m1.cpt("Patient")[{ 'Drug': 'Without', 'Gender': 'M' }] = [0.6, 0.4] #No Drug, Female -> healed in 0.4 of cases
m1.cpt("Patient")[{ 'Drug': 'With', 'Gender': 'F' }] = [0.3, 0.7] #Drug, Male -> healed 0.7 of cases
m1.cpt("Patient")[{ 'Drug': 'With', 'Gender': 'M' }] = [0.8, 0.2] #Drug, Female -> healed in 0.2 of cases
gmb.flow.row(m1, m1.cpt("Gender"), m1.cpt("Drug"), m1.cpt("Patient"))
```

`pyAgrum.fastBN(structure, domain_size=2)`

Create a Bayesian network with a dot-like syntax which specifies:

- the structure 'a->b->c;b->d<-e;',
- the type of the variables with different syntax (cf documentation).

Examples

```
>>> import pyAgrum as gum
>>> bn=gum.fastBN('A->B[1,3]<-C{yes|No}->D[2,4]<-E[1,2.5,3.9]',6)
```

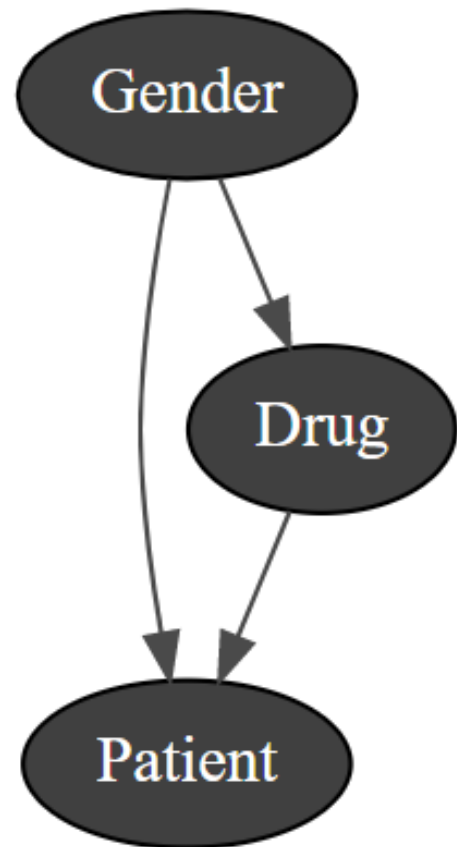
Parameters

- **structure** (*str*) – the string containing the specification
- **domain_size** (*int*) – the default domain size for variables

Returns

the resulting bayesian network

Return type

`pyAgrum.BayesNet`

Gender	
F	M
0.5000	0.5000

Gender	Drug	
	Without	With
F	0.2500	0.7500
M	0.7500	0.2500

Gender	Drug	Patient	
		Sick	Healed
F	Without	0.2000	0.8000
	With	0.3000	0.7000
M	Without	0.6000	0.4000
	With	0.8000	0.2000

<https://pyagrums.readthedocs.io/en/latest/index.html>

Simpson's paradox via pyAgrum

In a statistical study about a drug, we try to evaluate the latter's efficiency among a population of men and women.

Let's note: - Drug : drug taking - Patient : cured patient - Gender : patient's gender

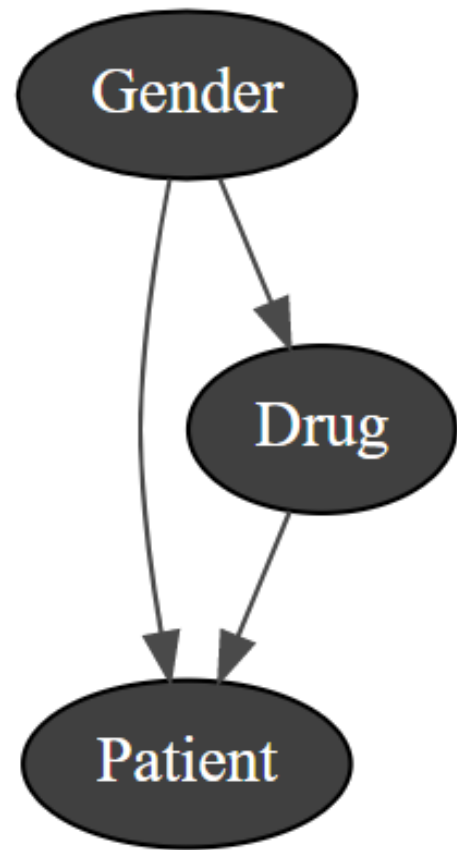
The model from the observed data is as follow :

```
m1 = gum.fastBN("Gender{F|M}->Drug{Without|With}->Patient{Sick|Healed}<-Gender")

m1.cpt("Gender")[:] = [0.5, 0.5]
m1.cpt("Drug")[:] = [[0.25, 0.75], #Gender=F
                    [0.75, 0.25]] #Gender=M

m1.cpt("Patient")[{ 'Drug': 'Without', 'Gender': 'F' }] = [0.2, 0.8] #No Drug, Male -> healed in 0.8 of cases
m1.cpt("Patient")[{ 'Drug': 'Without', 'Gender': 'M' }] = [0.6, 0.4] #No Drug, Female -> healed in 0.4 of cases
m1.cpt("Patient")[{ 'Drug': 'With', 'Gender': 'F' }] = [0.3, 0.7] #Drug, Male -> healed 0.7 of cases
m1.cpt("Patient")[{ 'Drug': 'With', 'Gender': 'M' }] = [0.8, 0.2] #Drug, Female -> healed in 0.2 of cases
gmb.flow.row(m1, m1.cpt("Gender"), m1.cpt("Drug"), m1.cpt("Patient"))
```

} Prepare the Conditional Probability Table



Gender	
F	M
0.5000	0.5000

Gender	Drug	
	Without	With
F	0.2500	0.7500
M	0.7500	0.2500

Gender	Drug	Patient	
		Sick	Healed
F	Without	0.2000	0.8000
	With	0.3000	0.7000
M	Without	0.6000	0.4000
	With	0.8000	0.2000

Simpson's paradox via pyAgrum

In a statistical study about a drug, we try to evaluate the latter's efficiency among a population of men and women.

Let's note: - Drug : drug taking - Patient : cured patient - Gender : patient's gender

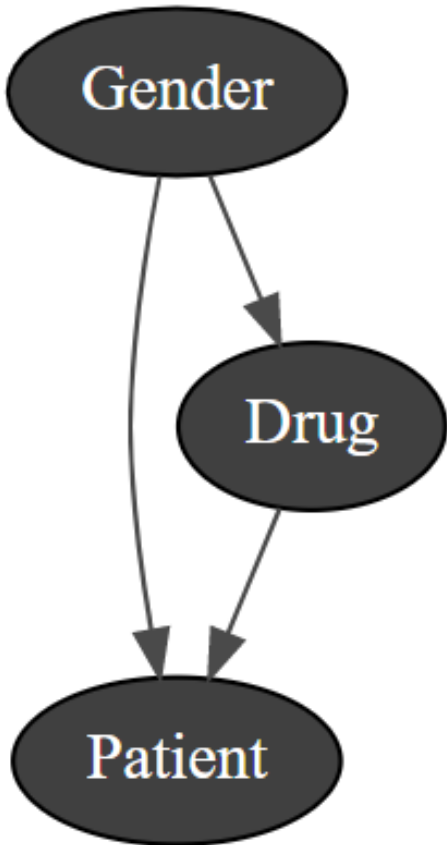
The model from the observed data is as follow :

```
m1 = gum.fastBN("Gender{F|M}->Drug{Without|With}->Patient{Sick|Healed}<-Gender")

m1.cpt("Gender")[:] = [0.5, 0.5]
m1.cpt("Drug")[:] = [[0.25, 0.75], #Gender=F
                    [0.75, 0.25]] #Gender=M

m1.cpt("Patient")[{ 'Drug': 'Without', 'Gender': 'F' }] = [0.2, 0.8] #No Drug, Male -> healed in 0.8 of cases
m1.cpt("Patient")[{ 'Drug': 'Without', 'Gender': 'M' }] = [0.6, 0.4] #No Drug, Female -> healed in 0.4 of cases
m1.cpt("Patient")[{ 'Drug': 'With', 'Gender': 'F' }] = [0.3, 0.7] #Drug, Male -> healed 0.7 of cases
m1.cpt("Patient")[{ 'Drug': 'With', 'Gender': 'M' }] = [0.8, 0.2] #Drug, Female -> healed in 0.2 of cases
gmb.flow.row(m1, m1.cpt("Gender"), m1.cpt("Drug"), m1.cpt("Patient"))
```

→ To display the CPT



Gender	
F	M
0.5000	0.5000

Gender	Drug	
	Without	With
F	0.2500	0.7500
M	0.7500	0.2500

Gender	Drug	Patient	
		Sick	Healed
F	Without	0.2000	0.8000
	With	0.3000	0.7000
M	Without	0.6000	0.4000
	With	0.8000	0.2000

<https://pyagrums.readthedocs.io/en/latest/index.html>

Simpson's paradox via pyAgrum



```
def getCuredObservedProba(m1, evs):
    evs0=dict(evs)
    evs1=dict(evs)
    evs0["Drug"]='Without'
    evs1["Drug"]='With'

    return gum.Potential().add(m1.variableFromName("Drug")).fillWith([
        gum.getPosterior(m1,target="Patient",evs=evs0)[1],
        gum.getPosterior(m1,target="Patient",evs=evs1)[1]
    ])

gnb.sideBySide(getCuredObservedProba(m1,{}),
    getCuredObservedProba(m1,{ 'Gender': 'F' }),
    getCuredObservedProba(m1,{ 'Gender': 'M' }),
    captions=["$P(Patient = Healed \mid Drug )$<br/>Taking $Drug$ is observed as efficient to cure",
        "$P(Patient = Healed \mid Gender=F,Drug)$<br/>except if the $gender$ of the patient is female",
        "$P(Patient = Healed \mid Gender=M,Drug)$<br/>... or male."])
```

Drug	
Without	With
0.5000	0.5750

*\$P(Patient = Healed \mid Drug)\$
Taking \$Drug\$ is observed as efficient to cure*

Drug	
Without	With
0.8000	0.7000

*\$P(Patient = Healed \mid Gender=F,Drug)\$
except if the \$gender\$ of the patient is female*

Drug	
Without	With
0.4000	0.2000

*\$P(Patient = Healed \mid Gender=M,Drug)\$
... or male.*

A Potential function is a function that associates a non-negative value (or probability) with each possible assignment of values to a set of random variables. Potential functions are used to represent the local relationships between random variables in a graphical model. Specifically, a potential function is associated with each factor node in the graph, which typically corresponds to a set of random variables in the model.

Simpson's paradox via pyAgrum



```
def getCuredObservedProba(m1, evs):
    evs0=dict(evs)
    evs1=dict(evs)
    evs0["Drug"]='Without'
    evs1["Drug"]='With'

    return gum.Potential().add(m1.variableFromName("Drug")).fillWith([
        gum.getPosterior(m1,target="Patient",evs=evs0)[1],
        gum.getPosterior(m1,target="Patient",evs=evs1)[1]
    ])

gnb.sideBySide(getCuredObservedProba(m1,{}),
    getCuredObservedProba(m1,{ 'Gender': 'F' }),
    getCuredObservedProba(m1,{ 'Gender': 'M' } ),
    captions=["$P(Patient = Healed \mid Drug )$<br/>Taking $Drug$ is observed as efficient to cure",
        "$P(Patient = Healed \mid Gender=F,Drug)$<br/>except if the $gender$ of the patient is female",
        "$P(Patient = Healed \mid Gender=M,Drug)$<br/>... or male."])
```

Drug	
Without	With
0.5000	0.5750

*\$P(Patient = Healed \mid Drug)\$
Taking \$Drug\$ is observed as efficient to cure*

Drug	
Without	With
0.8000	0.7000

*\$P(Patient = Healed \mid Gender=F,Drug)\$
except if the \$gender\$ of the patient is female*

Drug	
Without	With
0.4000	0.2000

*\$P(Patient = Healed \mid Gender=M,Drug)\$
... or male.*

pyAgrum.getPosterior() is a function from the Python package pyAgrum which is used to compute the posterior probabilities of a set of variables given some evidence. The function returns an array of values because it is designed to compute the posterior probability distribution of the variables, which is a probability distribution over all possible values of the variables.

Simpson's paradox via pyAgrum



```
def getCuredObservedProba(m1, evs):
    evs0=dict(evs)
    evs1=dict(evs)
    evs0["Drug"]='Without'
    evs1["Drug"]='With'

    return gum.Potential().add(m1.variableFromName("Drug")).fillWith([
        gum.getPosterior(m1,target="Patient",evs=evs0)[1],
        gum.getPosterior(m1,target="Patient",evs=evs1)[1]
    ])

gnb.sideBySide(getCuredObservedProba(m1,{}),
    getCuredObservedProba(m1,{ 'Gender': 'F' }),
    getCuredObservedProba(m1,{ 'Gender': 'M' }),
    captions=["$P(Patient = Healed \mid Drug )$<br/>Taking $Drug$ is observed as efficient to cure",
        "$P(Patient = Healed \mid Gender=F,Drug)$<br/>except if the $gender$ of the patient is female",
        "$P(Patient = Healed \mid Gender=M,Drug)$<br/>... or male."])
```

Drug	
Without	With
0.5000	0.5750

*\$P(Patient = Healed \mid Drug)\$
Taking \$Drug\$ is observed as efficient to cure*

Drug	
Without	With
0.8000	0.7000

*\$P(Patient = Healed \mid Gender=F,Drug)\$
except if the \$gender\$ of the patient is female*

Drug	
Without	With
0.4000	0.2000

*\$P(Patient = Healed \mid Gender=M,Drug)\$
... or male.*

Those results form a paradox called Simpson paradox :

$P(C|\neg D) = 0.5 < P(C|D) = 0.575$

$P(C|\neg D, G=Male) = 0.8 > P(C|D, G=Male)=0.7$

$P(C|\neg D, G=Female) = 0.4 > P(C|D, G=Female)=0.2$

<https://pyagrum.readthedocs.io/en/latest/index.html>

Simpson's paradox via pyAgrum



```
def getCuredObservedProba(m1, evs):
    evs0=dict(evs)
    evs1=dict(evs)
    evs0["Drug"]='Without'
    evs1["Drug"]='With'

    return gum.Potential().add(m1.variableFromName("Drug")).fillWith([
        gum.getPosterior(m1,target="Patient",evs=evs0)[1],
        gum.getPosterior(m1,target="Patient",evs=evs1)[1]
    ])

gnb.sideBySide(getCuredObservedProba(m1,{ } ),
    getCuredObservedProba(m1,{ 'Gender': 'F' } ),
    getCuredObservedProba(m1,{ 'Gender': 'M' } ),
    captions=["$P(Patient = Healed \mid Drug )$<br/>Taking $Drug$ is observed as efficient to cure",
        "$P(Patient = Healed \mid Gender=F,Drug)$<br/>except if the $gender$ of the patient is female",
        "$P(Patient = Healed \mid Gender=M,Drug)$<br/>... or male."])
```

Drug	
Without	With
0.5000	0.5750

*$P(\text{Patient} = \text{Healed} \mid \text{Drug})$
Taking Drug is observed as efficient to cure*

Drug	
Without	With
0.8000	0.7000

*$P(\text{Patient} = \text{Healed} \mid \text{Gender}=\text{F}, \text{Drug})$
except if the gender of the patient is female*

Drug	
Without	With
0.4000	0.2000

*$P(\text{Patient} = \text{Healed} \mid \text{Gender}=\text{M}, \text{Drug})$
... or male.*

Actually, giving a drug is not an **observation in our model but rather an intervention**.
What if we use intervention instead of observation ?

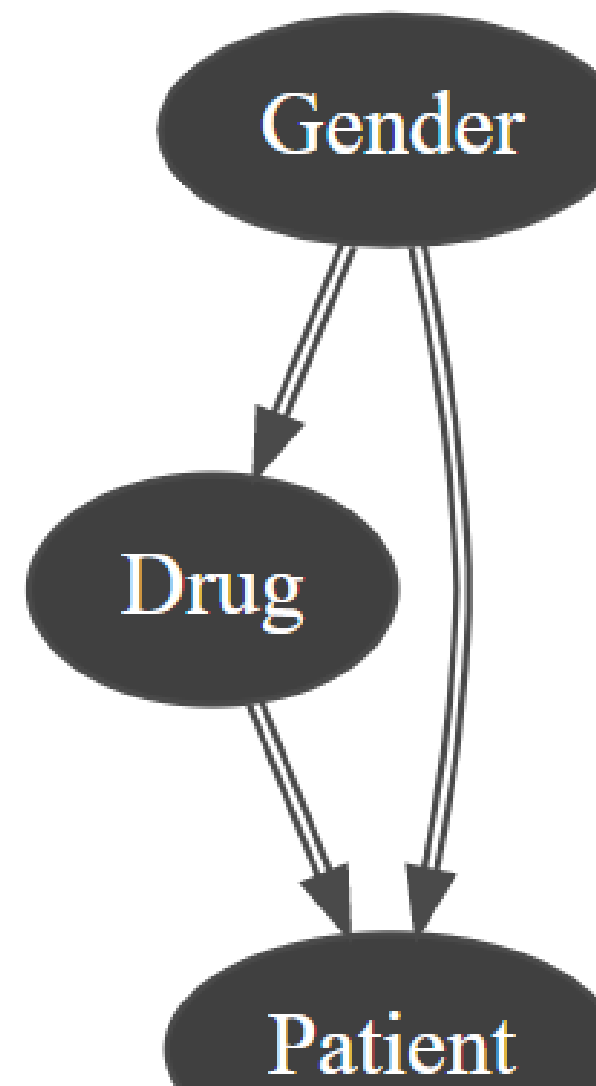
<https://pyagrument.readthedocs.io/en/latest/index.html>

Simpson's paradox via pyAgrum



How to compute causal impacts on the patient's health ?
We propose this causal model.

```
d1 = cs1.CausalModel(m1)
cs1nb.showCausalModel(d1)
```



```
class pyAgrum.causal.CausalModel (bn: pyAgrum.BayesNet, latentVarsDescriptor: Optional[List[Tuple[str, Tuple[str, str]]]] = None, keepArcs: bool = False)
```

From an observational BNs and the description of latent variables, this class represent a complete causal model obtained by adding the latent variables specified in `latentVarsDescriptor` to the Bayesian network `bn`.

Parameters:

- `bn` – a observational Bayesian network
- `latentVarsDescriptor` – list of couples (<latent variable name>, <list of affected variables' ids>).
- `keepArcs` – By default, the arcs between variables affected by a common latent variable will be removed but this can be avoided by setting `keepArcs` to `True`

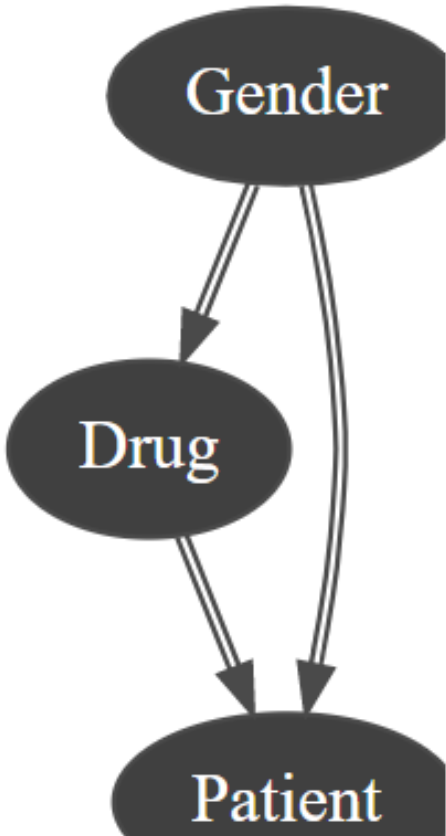
<https://pyagrums.readthedocs.io/en/latest/index.html>

Simpson's paradox via pyAgrum

How to compute causal impacts on the patient's health ?

Computing $P(\text{Patient}=\text{Healed}|\hookrightarrow\text{Drug}=\text{Without})$

```
cslnb.showCausalImpact(d1, "Patient", doing="Drug", values={"Drug" : "Without"})
```



Causal Model

```
pyAgrum.causal.notebook.showCausalImpact (model: pyAgrum.causal._CausalModel.CausalModel, on: Union[str, Set[str]], doing: Union[str, Set[str]], knowing: Optional[Set[str]] = None, values: Optional[Dict[str, int]] = None)
```

display a HTML representing of the three values defining a causal impact : formula, value, explanation :param model: the causal model :param on: the impacted variable(s) :param doing: the variable(s) of intervention :param knowing: the variable(s) of evidence :param values : values for certain variables

$$P(\text{Patient} \mid \hookrightarrow \text{Drug}) = \sum_{\text{Gender}} \{P(\text{Patient} \mid \text{Drug}, \text{Gender}) \cdot P(\text{Gender})\}$$

Explanation : backdoor ["Gender"] found.

Patient	
Sick	Healed
0.4000	0.6000

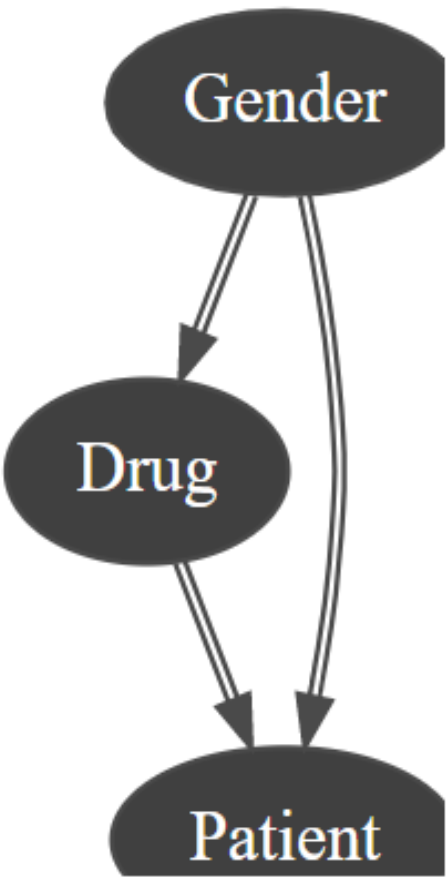
Impact

Simpson's paradox via pyAgrum

How to compute causal impacts on the patient's health ?

Computing $P(\text{Patient}=\text{Healed}|\hookrightarrow\text{Drug}=\text{With})$

```
d1 = cs1.CausalModel(m1)
cslnb.showCausalImpact(d1, "Patient", "Drug", values={"Drug" : "With"})
```



Causal Model

$$P(\text{Patient} \mid \hookrightarrow \text{Drug}) = \sum_{\text{Gender}} \{P(\text{Patient} \mid \text{Drug}, \text{Gender}) \cdot P(\text{Gender})\}$$

Explanation : backdoor [Gender] found.

Patient	
Sick	Healed
0.5500	0.4500

Impact

Simpson's paradox via pyAgrum



And then : $P(Patient = Healed \mid \hookrightarrow Drug = With) = 0.45$

Therefore : $P(Patient = Healed \mid \hookrightarrow Drug = Without) = 0.6 > P(Patient = Healed \mid \hookrightarrow Drug = With) = 0.45$

Which means that taking this drug would not enhance the patient's healing process, and it is better not to prescribe this drug for treatment.



<https://pyagrums.readthedocs.io/en/latest/index.html>

Questions

