WPMSIIP 2014

credal networks under epistemic irrelevance

Jasper De Bock

11 September 2014, Gent

Research group

SYSTeMS

UNIVERSITEIT
GENT

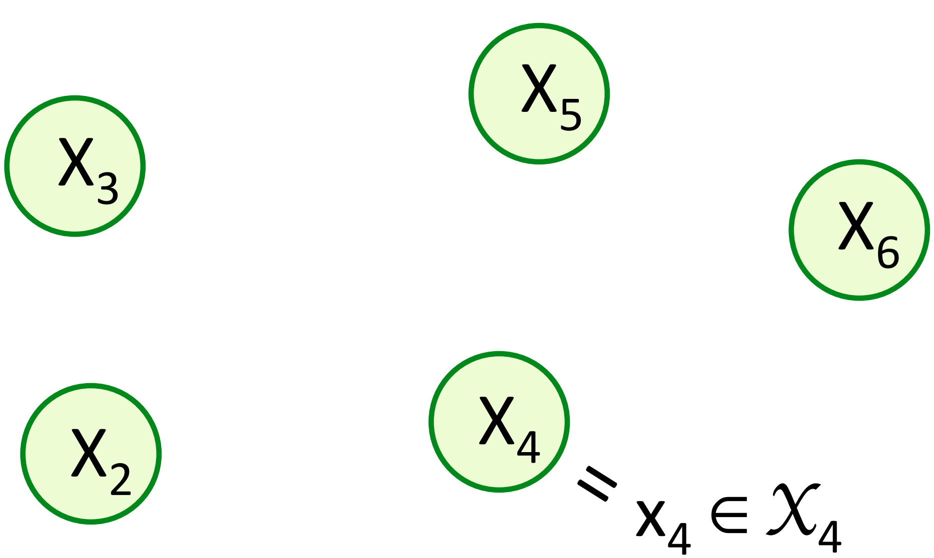
Stavros Lopatatzidis

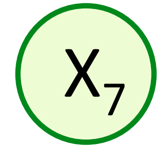
Arthur Van Camp

Jasper De Bock

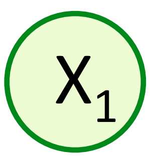
Gert de Cooman

Credal networks: basic setup



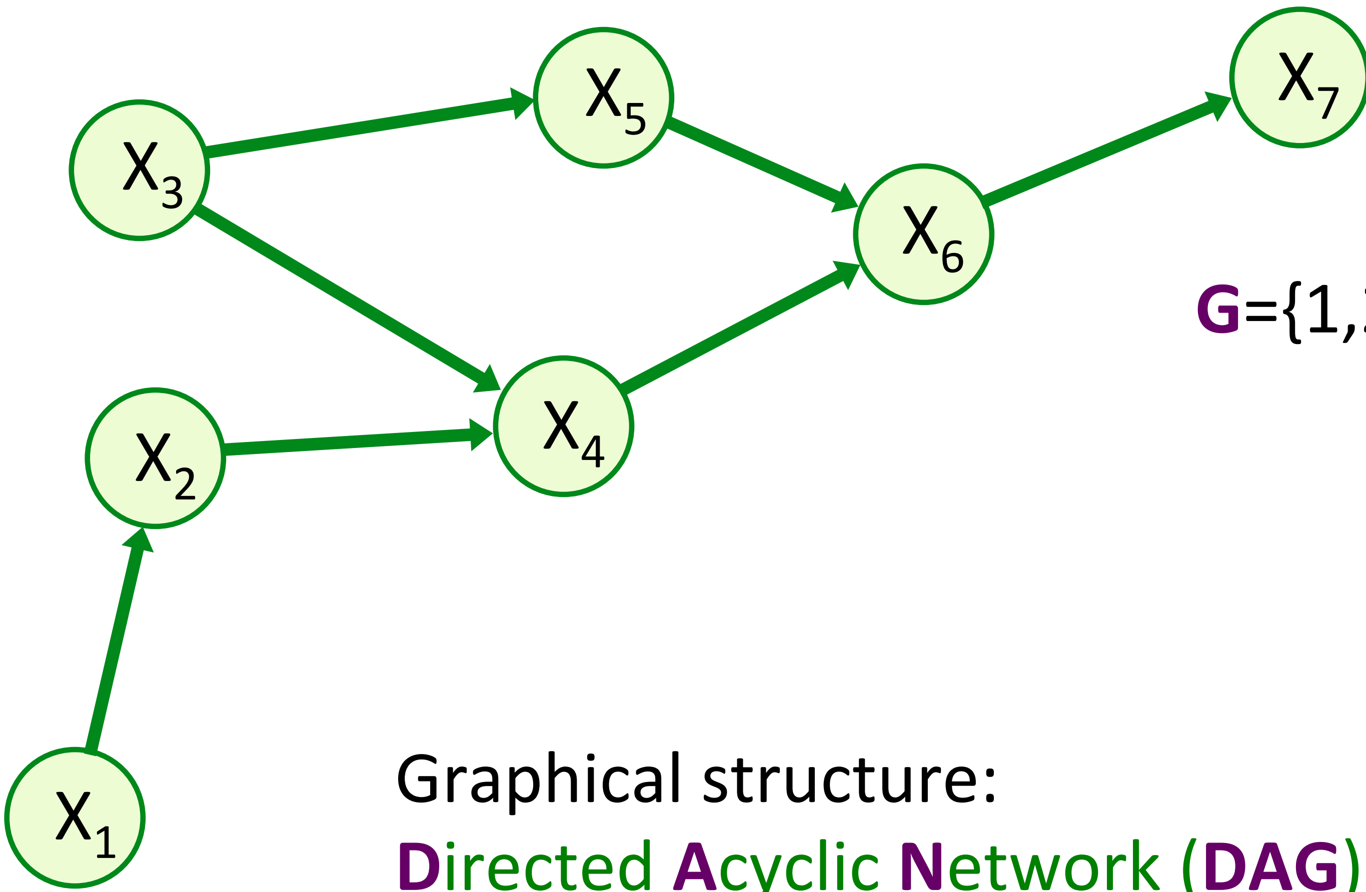


A finite number of **variables**



Every **variable X_s** takes values x_s in some finite non-empty set $\mathcal{X}_s$

Credal networks: basic setup

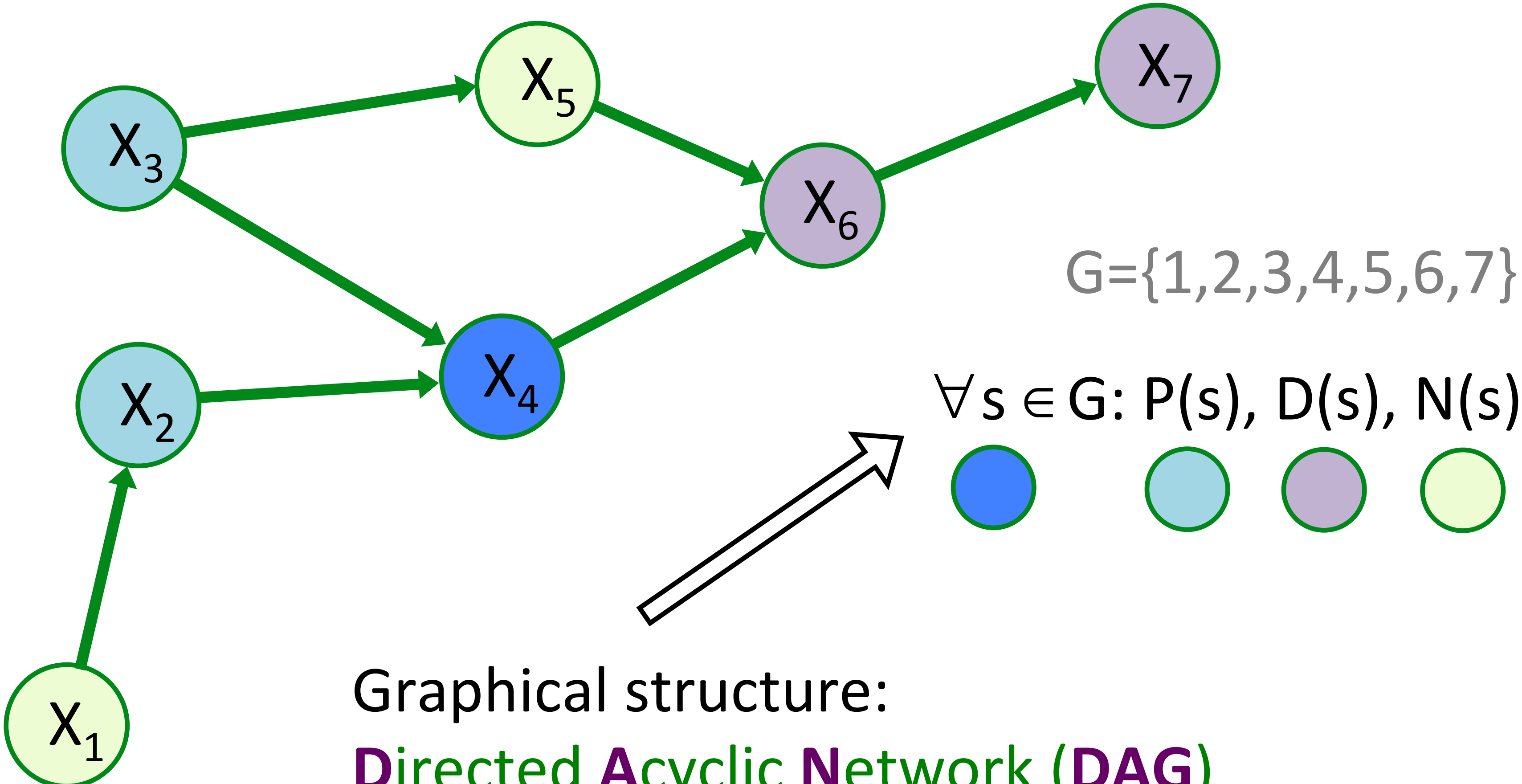


G={1,2,3,4,5,6,7}

Graphical structure:

Directed **A**cyclic **N**etwork (**DAG**)

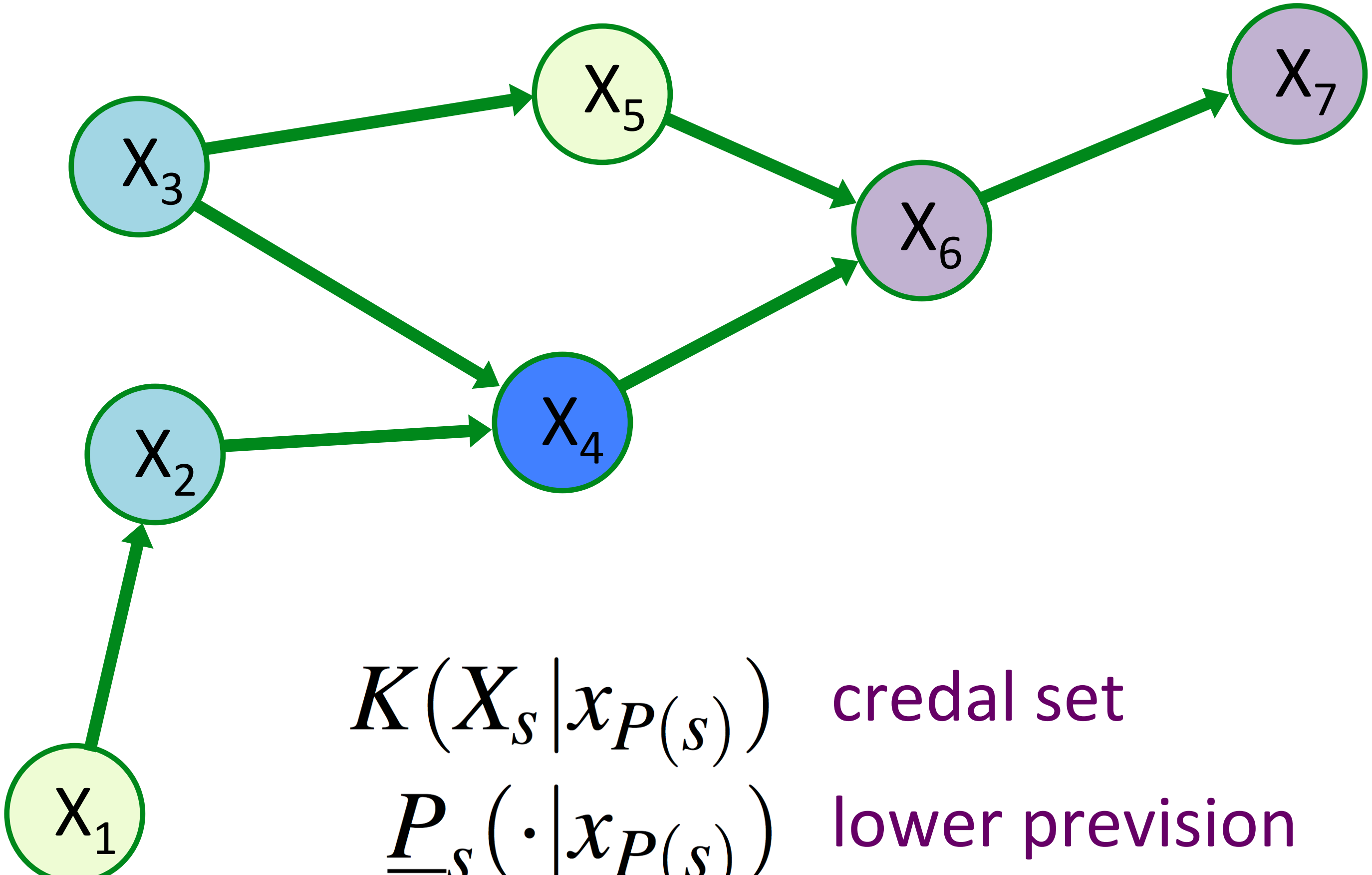
Credal networks: basic setup



Graphical structure:

Directed **A**cyclic **N**etwork (**DAG**)

Credal networks: local uncertainty models

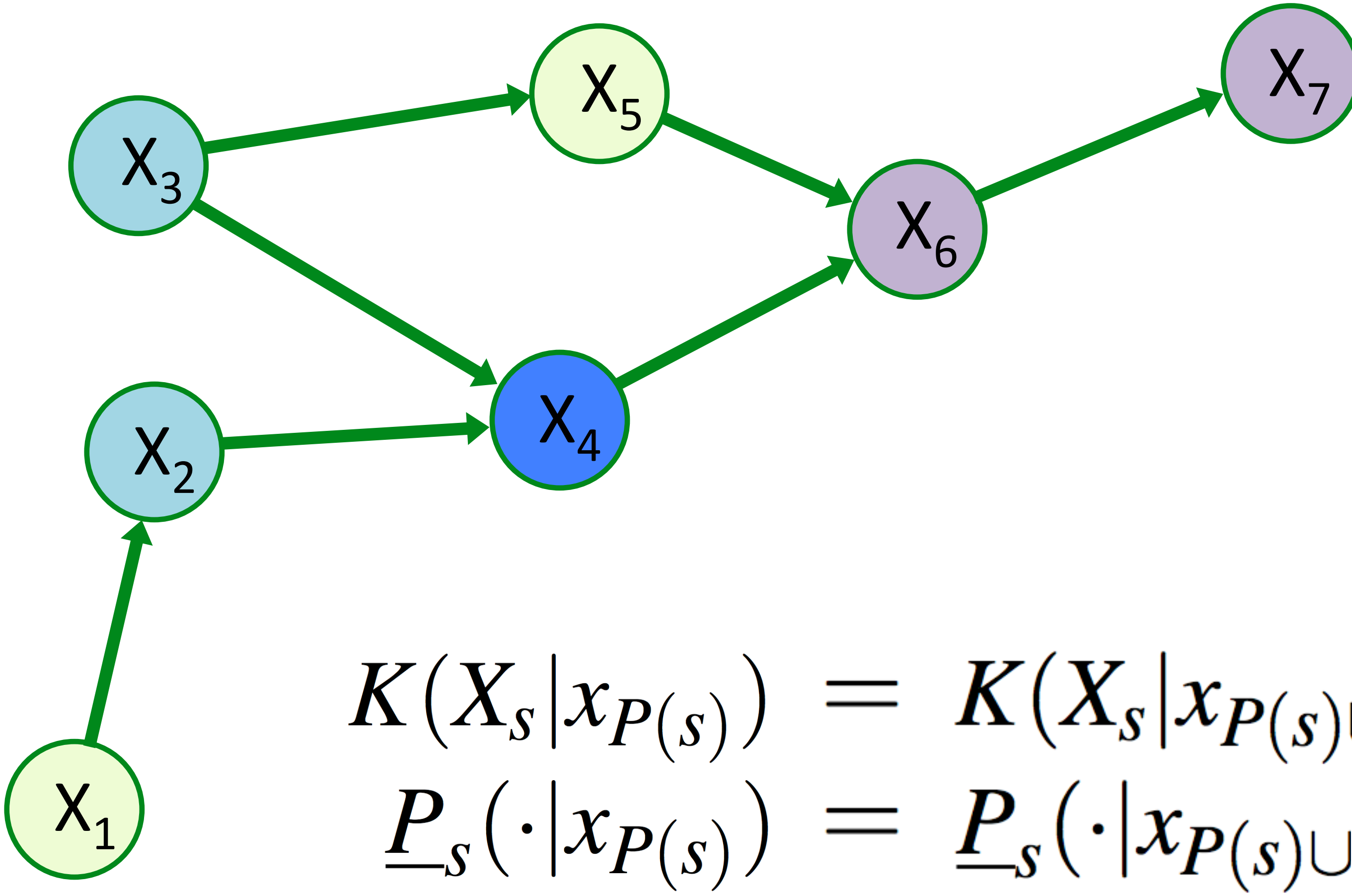


$K(X_s | x_{P(s)})$ credal set

$\underline{P}_s(\cdot | x_{P(s)})$ lower prevision

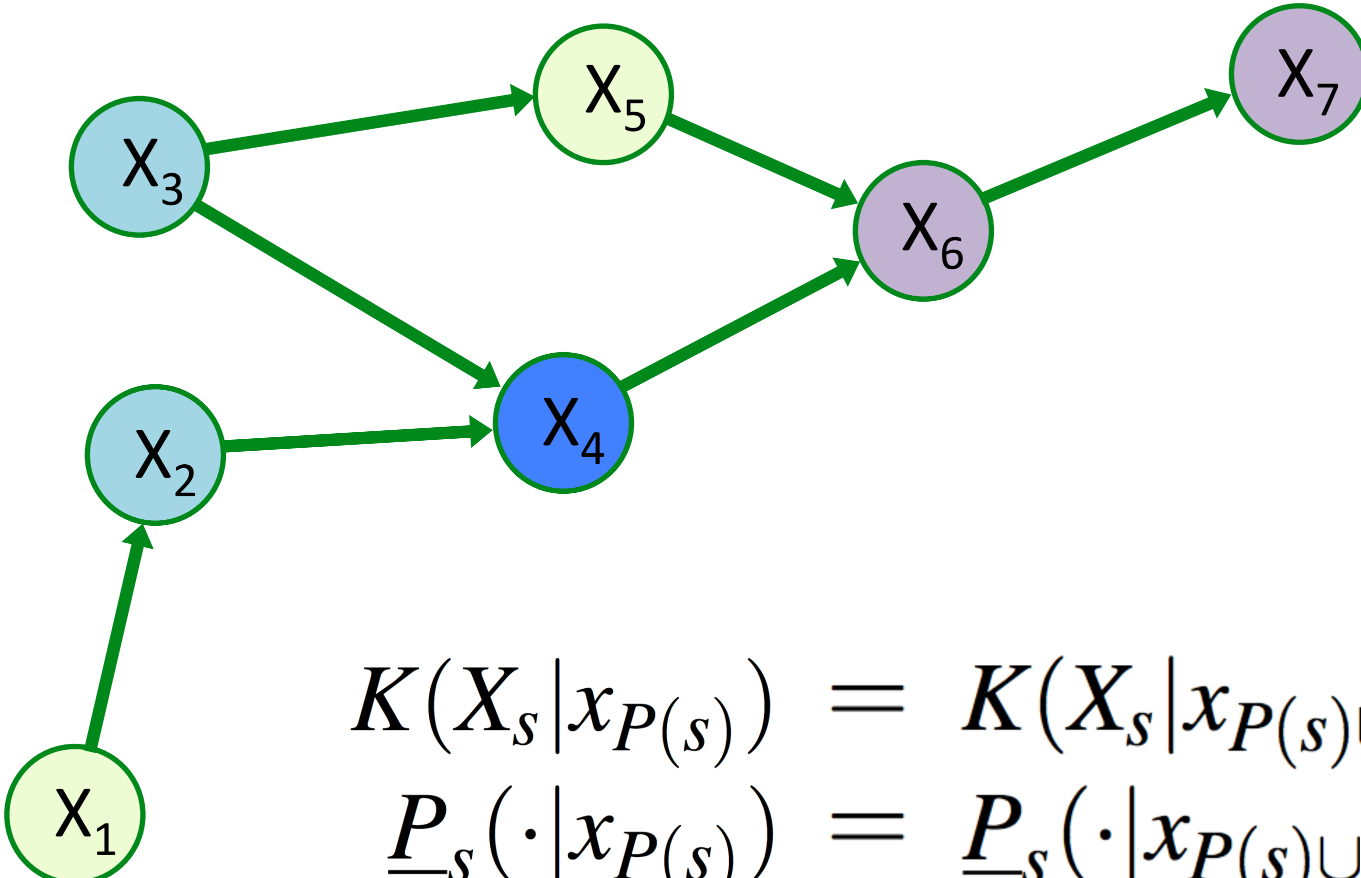
$\mathscr{D}_{s \rfloor x_{P(s)}}$ set of desirable gambles

Credal networks: epistemic irrelevance



$$K(X_s|x_{P(s)}) = K(X_s|x_{P(s)\cup N(s)})$$
$$\underline{P}_s(\cdot|x_{P(s)}) = \underline{P}_s(\cdot|x_{P(s)\cup N(s)})$$
$$\mathscr{D}_{s\rfloor x_{P(s)}} = \mathscr{D}_{s\rfloor x_{P(s)\cup N(s)}}$$

Credal networks: a joint model

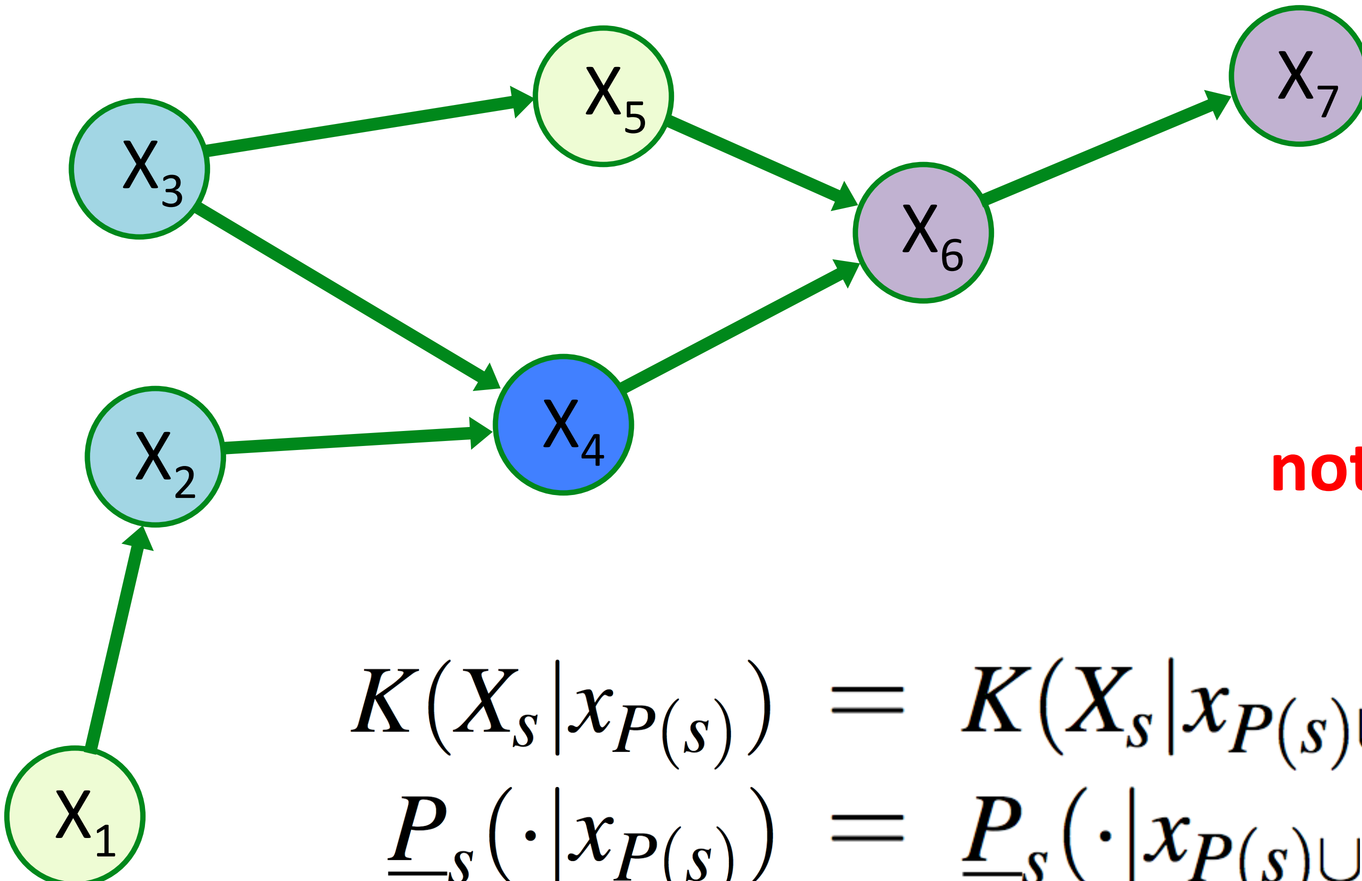


$$K(X_G)$$
$$\underline{P}_G$$
$$\mathscr{D}_G$$

?

$$K(X_s|x_{P(s)}) = K(X_s|x_{P(s)\cup N(s)})$$
$$\underline{P}_s(\cdot|x_{P(s)}) = \underline{P}_s(\cdot|x_{P(s)\cup N(s)})$$
$$\mathscr{D}_s\rfloor x_{P(s)} = \mathscr{D}_s\rfloor x_{P(s)\cup N(s)}$$

Credal networks: a joint model

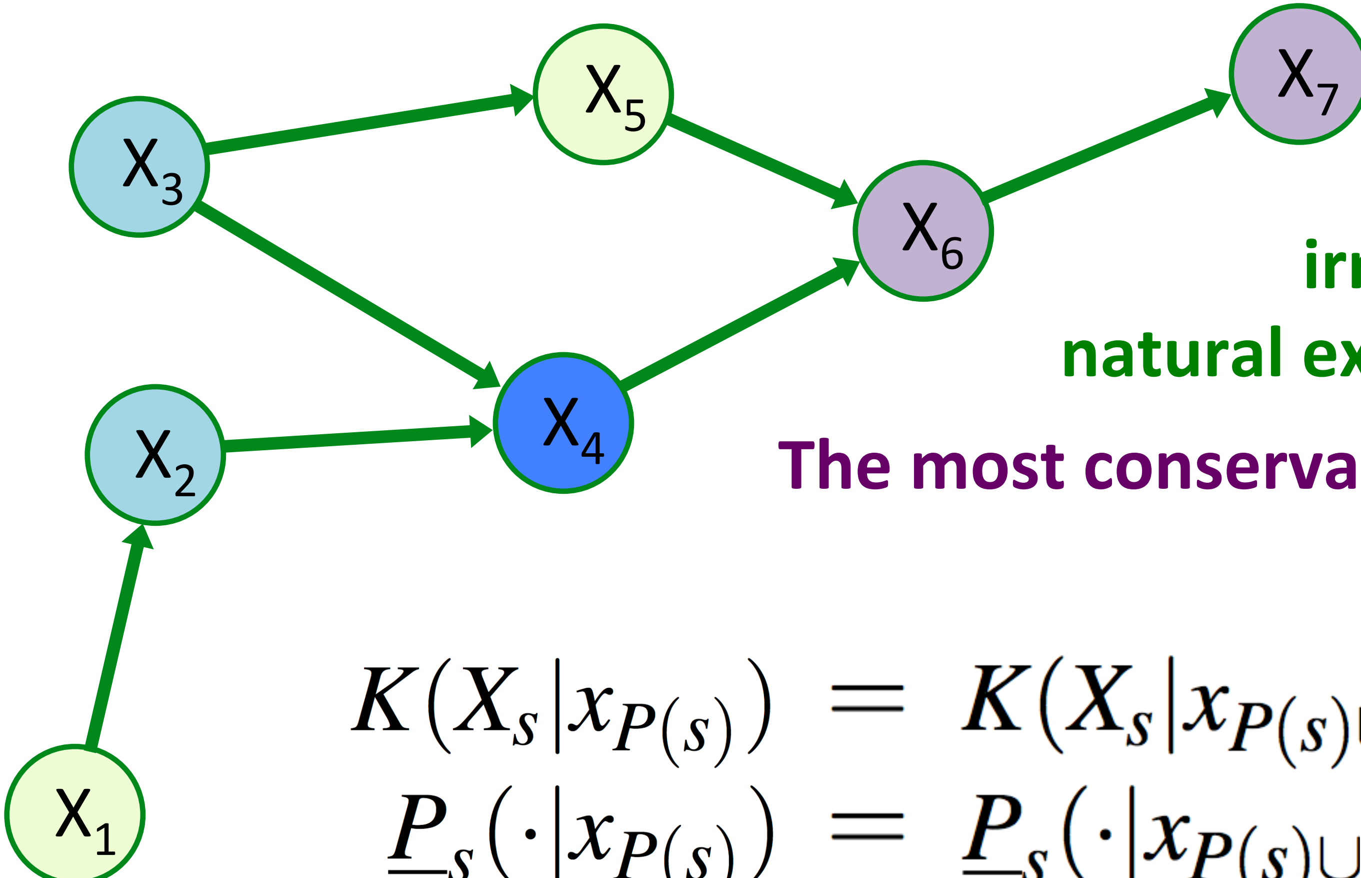


$$K(X_G)$$
$$\underline{P}_G$$
$$\mathscr{D}_G$$

not unique !

$$K(X_s|x_{P(s)}) = K(X_s|x_{P(s)\cup N(s)})$$
$$\underline{P}_s(\cdot|x_{P(s)}) = \underline{P}_s(\cdot|x_{P(s)\cup N(s)})$$
$$\mathscr{D}_{s\rfloor x_{P(s)}} = \mathscr{D}_{s\rfloor x_{P(s)\cup N(s)}}$$

Credal networks: a joint model



$K^{\mathrm{irr}}(X_G)$

$\underline{P}^{\mathrm{irr}}_G$

$\mathscr{D}^{\mathrm{irr}}_G$

irrelevant natural extension

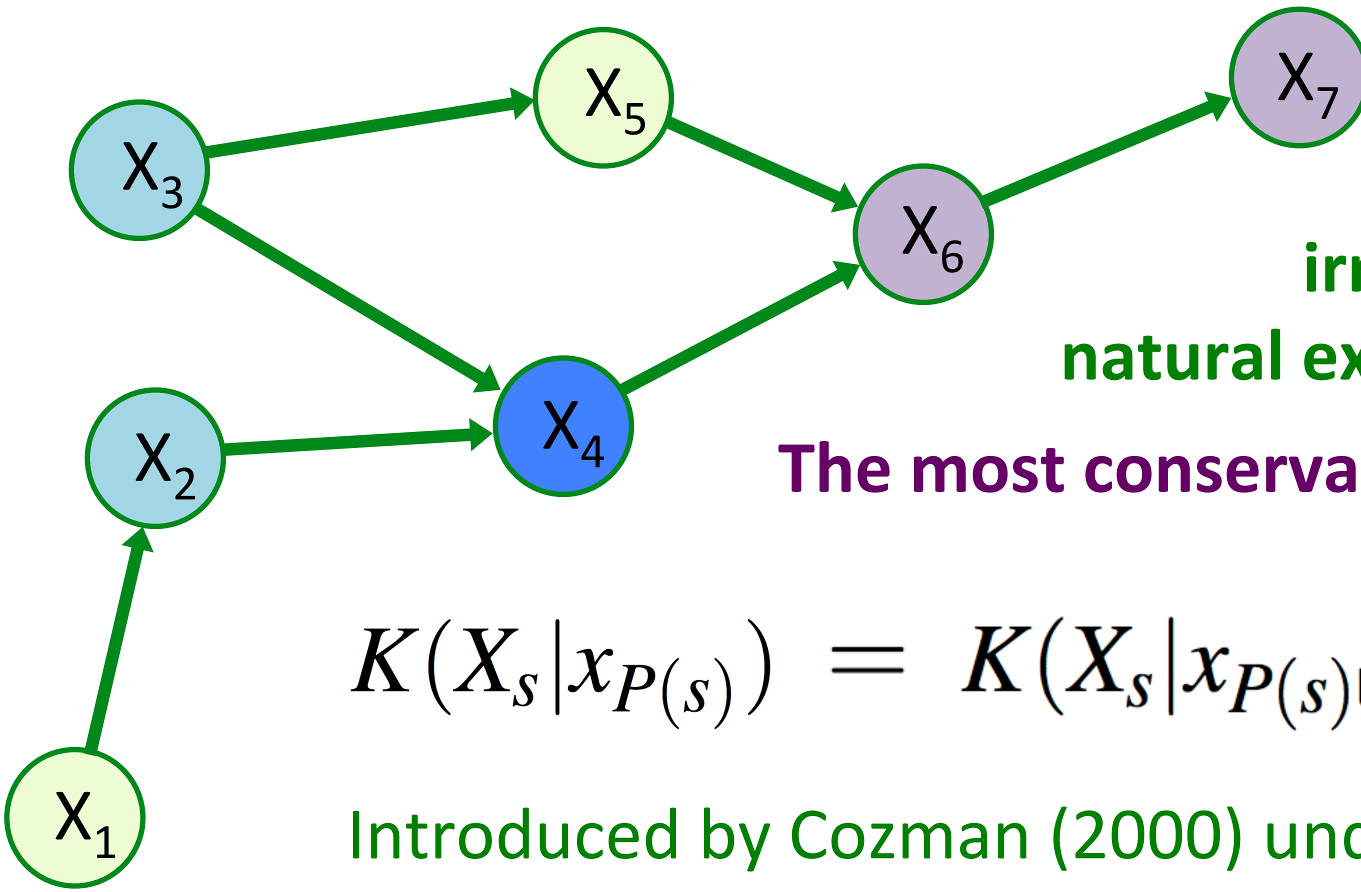
The most conservative one !

$$K(X_s|x_{P(s)}) = K(X_s|x_{P(s)\cup N(s)})$$

$$\underline{P}_s(\cdot|x_{P(s)}) = \underline{P}_s(\cdot|x_{P(s)\cup N(s)})$$

$$\mathscr{D}_{s\rfloor x_{P(s)}} = \mathscr{D}_{s\rfloor x_{P(s)\cup N(s)}}$$

Credal networks using credal sets



$K^{\text{irr}}(X_G)$

irrelevant natural extension

The most conservative one !

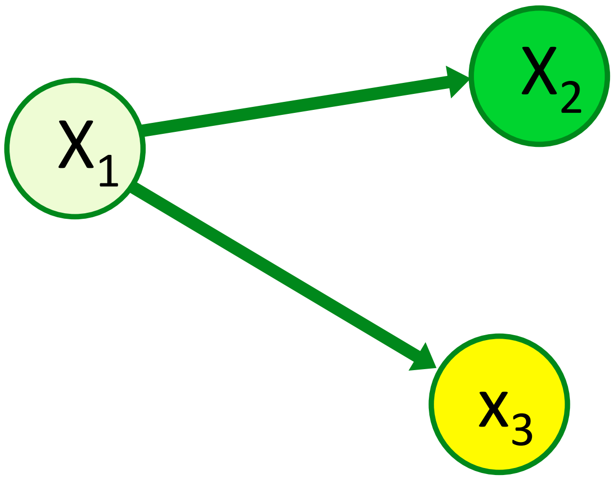
$$K(X_s|x_{P(s)}) = K(X_s|x_{P(s)\cup N(s)})$$

Introduced by Cozman (2000) under the assumption of positive lower probability.

Description in terms of linear constraints!

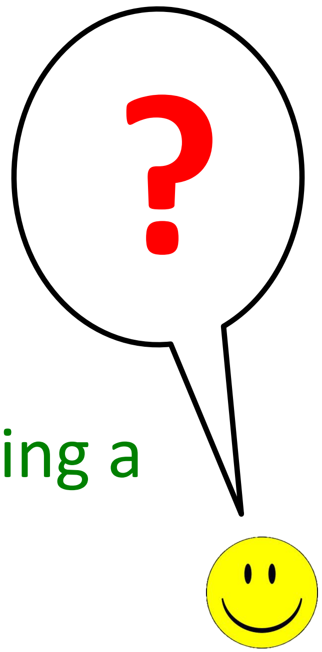
Credal networks using credal sets

$$K^{\text{irr}}(X_G)$$



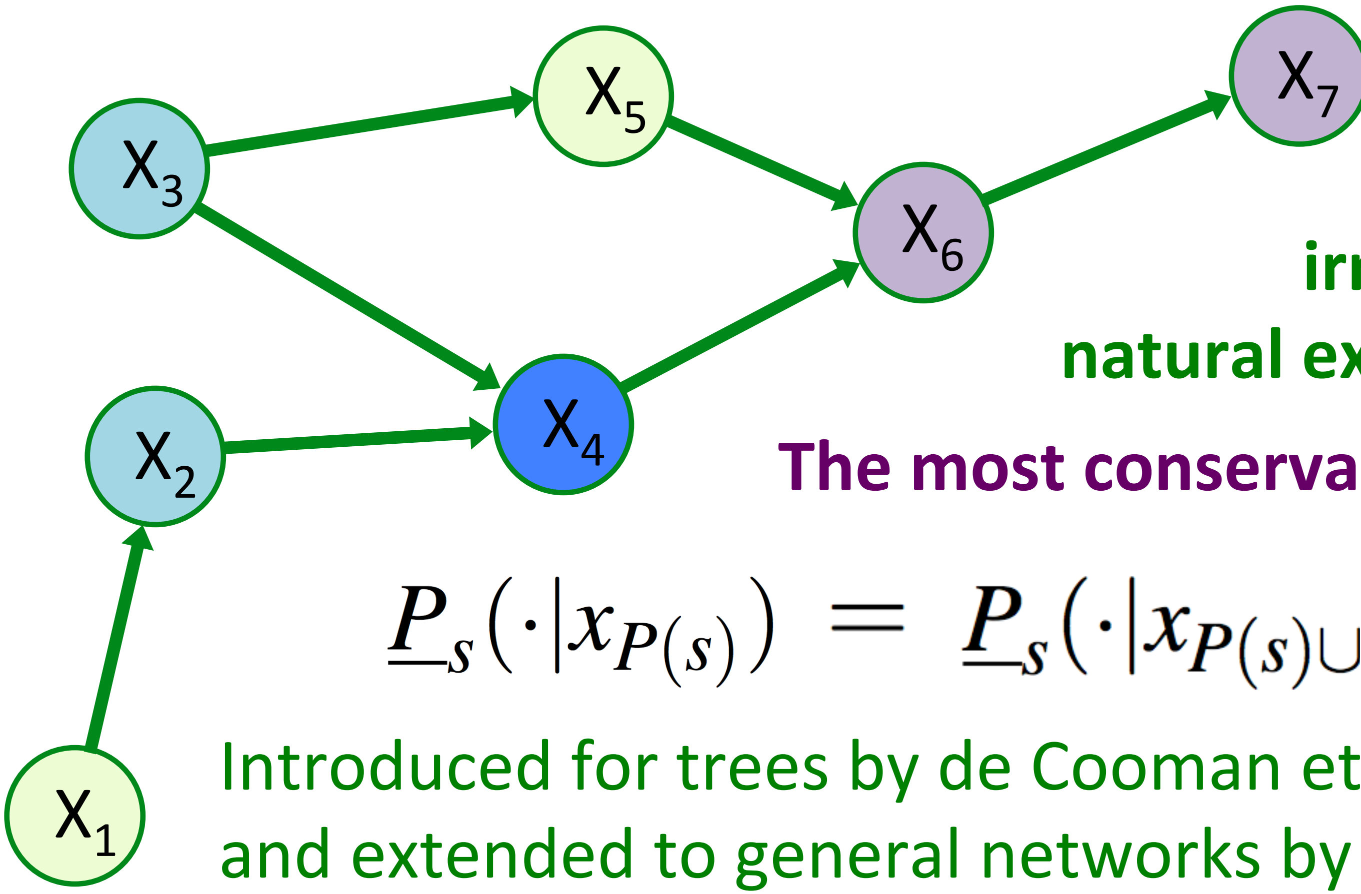
Query nodes

Evidence nodes



- Inference problems can be reduced to solving a (potentialy large) linear program!
- Lots of potential to derive both outer and inner approximations

Credal networks using lower previsions



$\underline{P}_G^{\mathrm{irr}}$

irrelevant natural extension

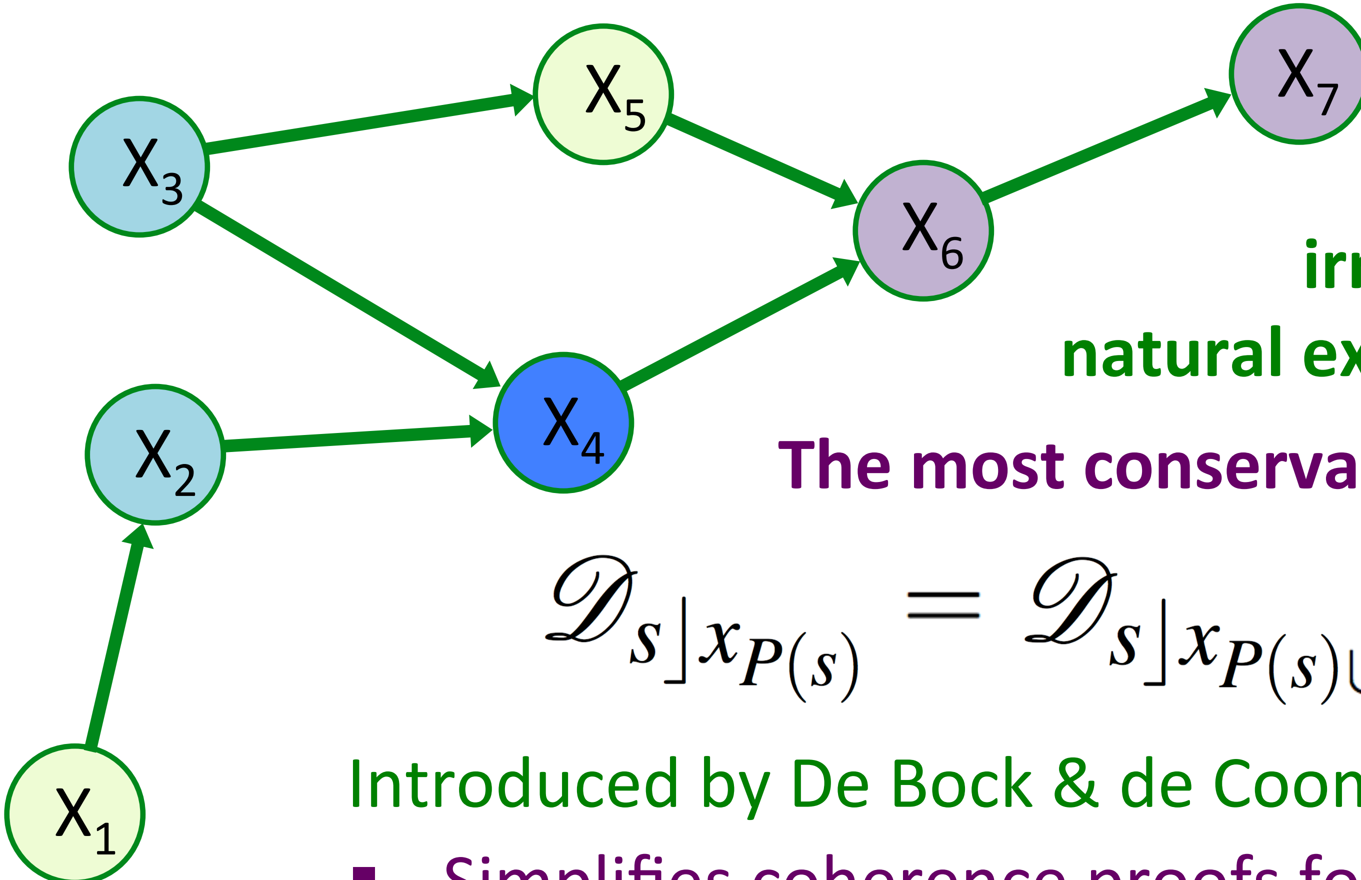
The most conservative one !

$$\underline{P}_s(\cdot|x_{P(s)}) = \underline{P}_s(\cdot|x_{P(s)\cup N(s)})$$

Introduced for trees by de Cooman et al. (2010) and extended to general networks by De Bock & de Cooman (2013), without positivity assumptions.

The joint is still described by the same linear constraints!

Credal networks using SDGs



$$\mathscr{D}_G^{\mathrm{irr}}$$

irrelevant natural extension

The most conservative one !

$$\mathscr{D}_{s \rfloor x_{P(s)}} = \mathscr{D}_{s \rfloor x_{P(s) \cup N(s)}}$$

Introduced by De Bock & de Cooman (2013)

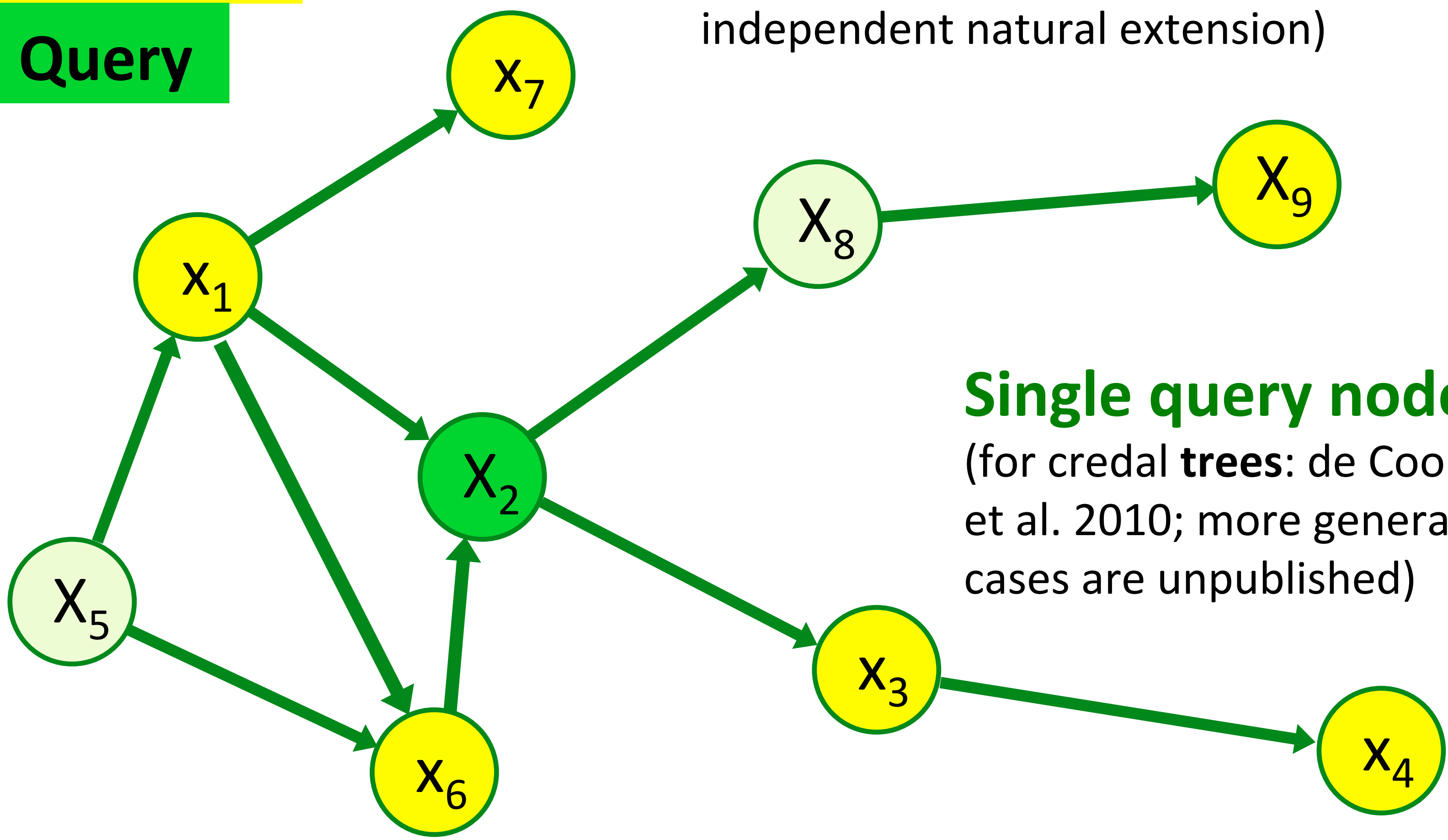
- Simplifies coherence proofs for LPs
- (conditional) marginalisation properties
- AD-separation implies irrelevance

efficient algorithms (linear for trees!)

Evidence

Query

(the joint can be constructed recursively by applying marginal extension and independent natural extension)



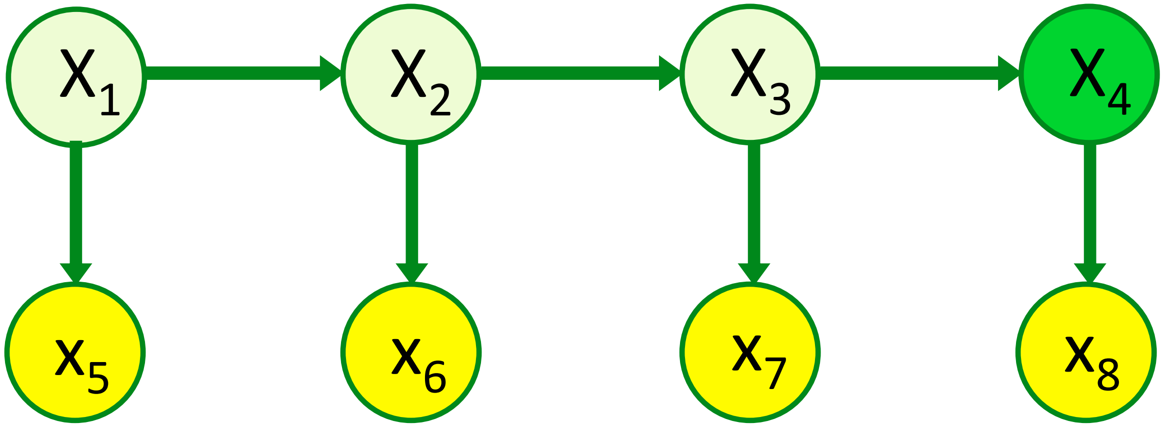
Single query node

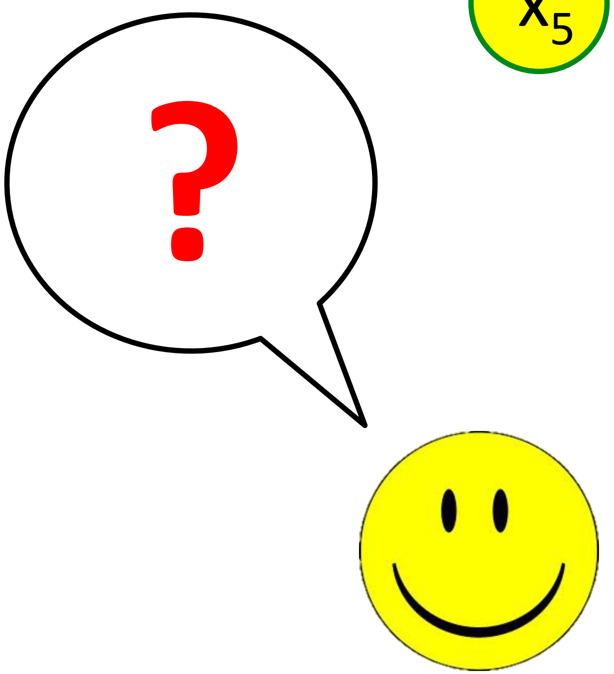
(for credal **trees**: de Cooman et al. 2010; more general cases are unpublished)

efficient algorithms (linear for trees!)

Evidence

Query





Single query node

(de Cooman et al. 2010)

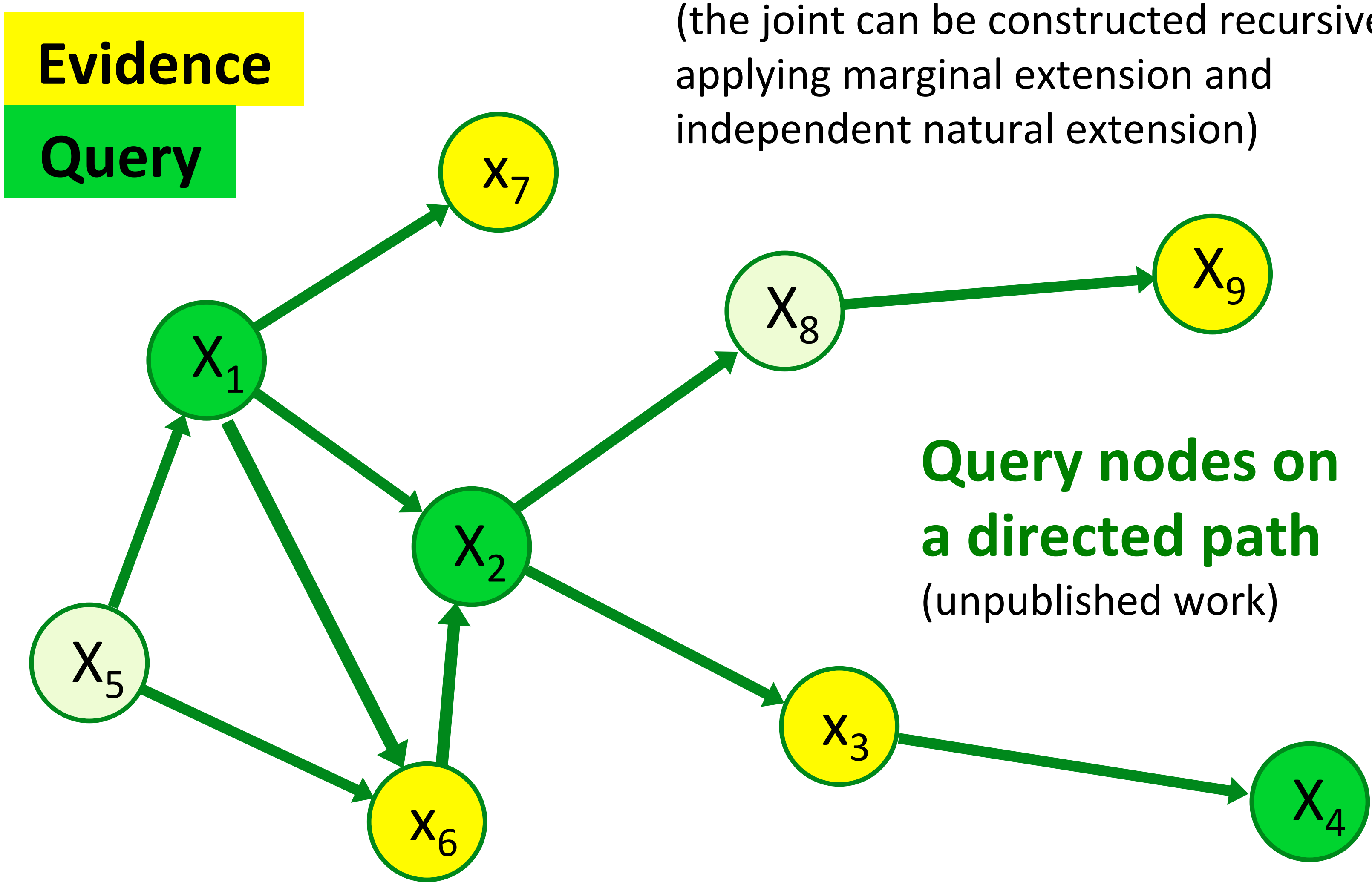
Example:

kalman filtering

(Benavoli et al. 2011)

efficient algorithms (linear for trees!)

(the joint can be constructed recursively by applying marginal extension and independent natural extension)

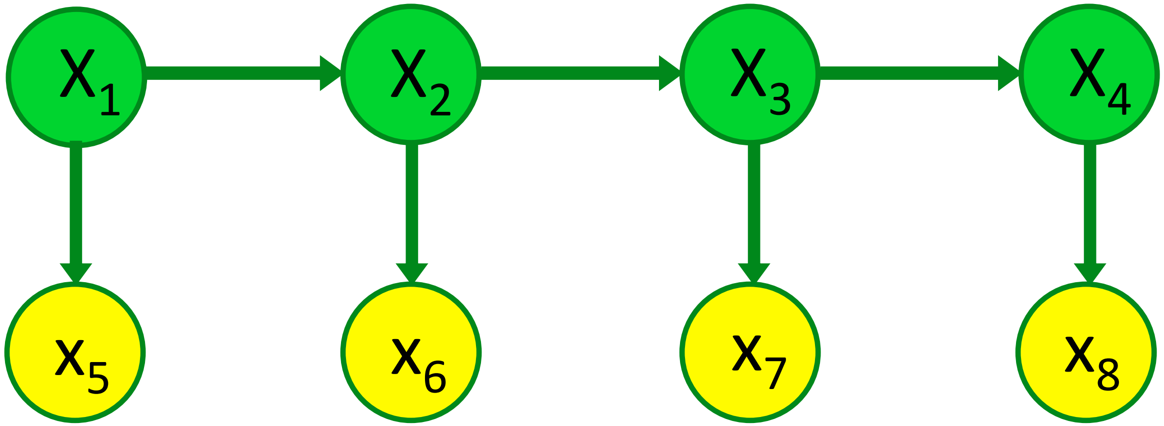


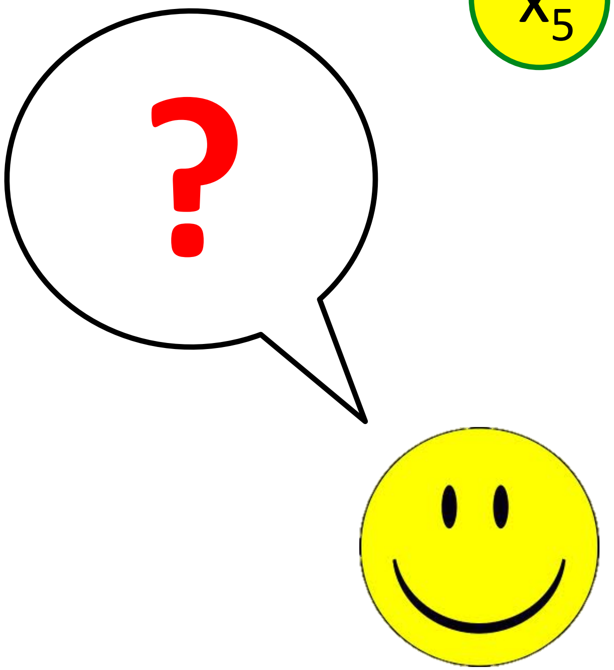
Query nodes on a directed path
(unpublished work)

efficient algorithms (linear for trees!)

Evidence

Query





Query nodes on a directed path

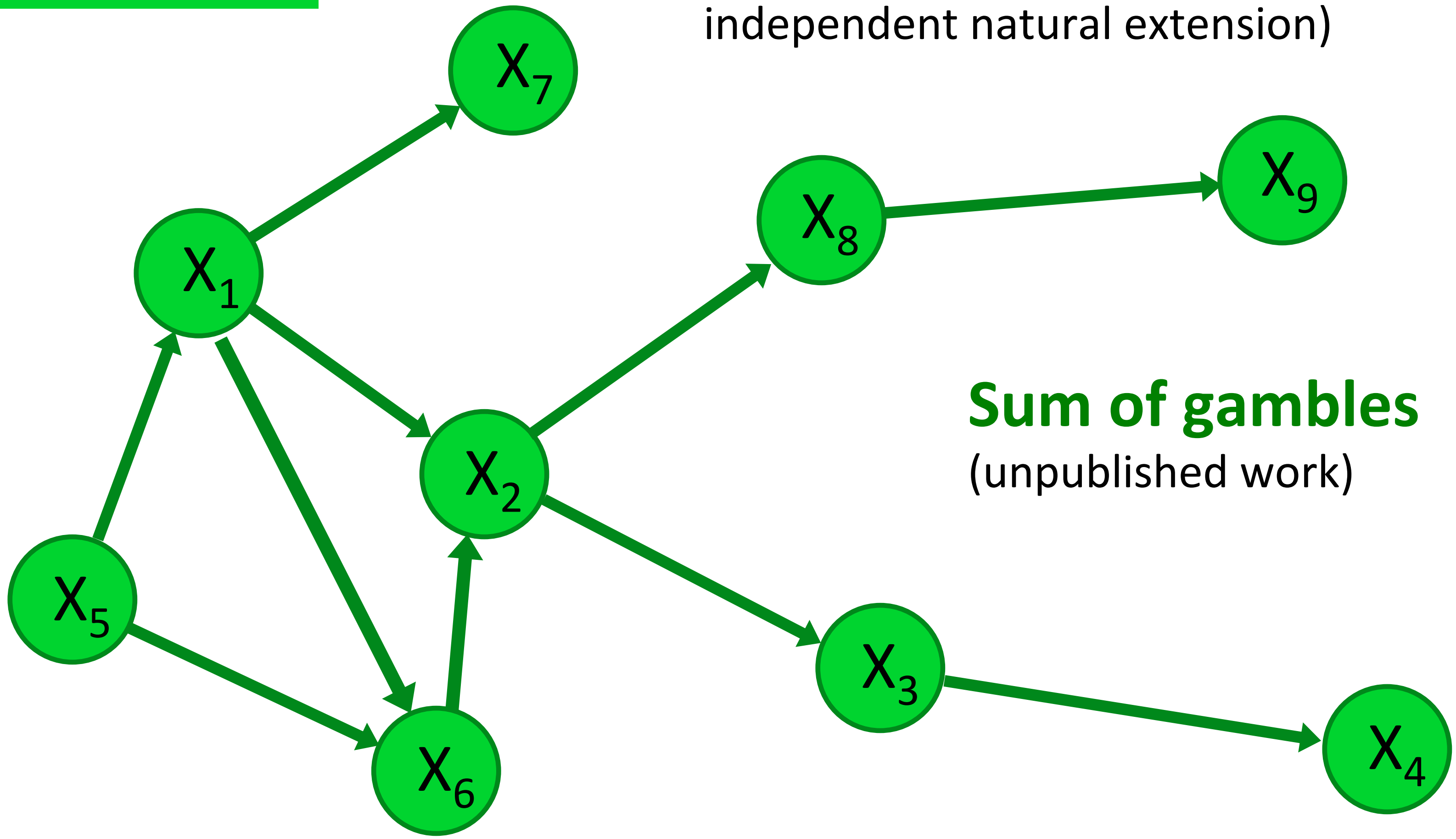
(unpublished work)

Example: finding maximin solution in imprecise Viterbi

efficient algorithms (linear for trees!)

Gambles

(the joint can be constructed recursively by applying marginal extension and independent natural extension)

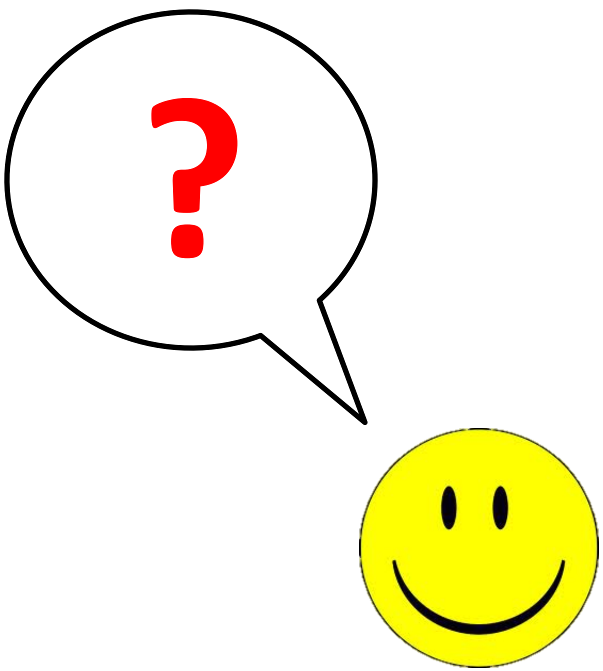


Sum of gambles

(unpublished work)

efficient algorithms (linear for trees!)

Gambles



Sum of gambles

(unpublished work)

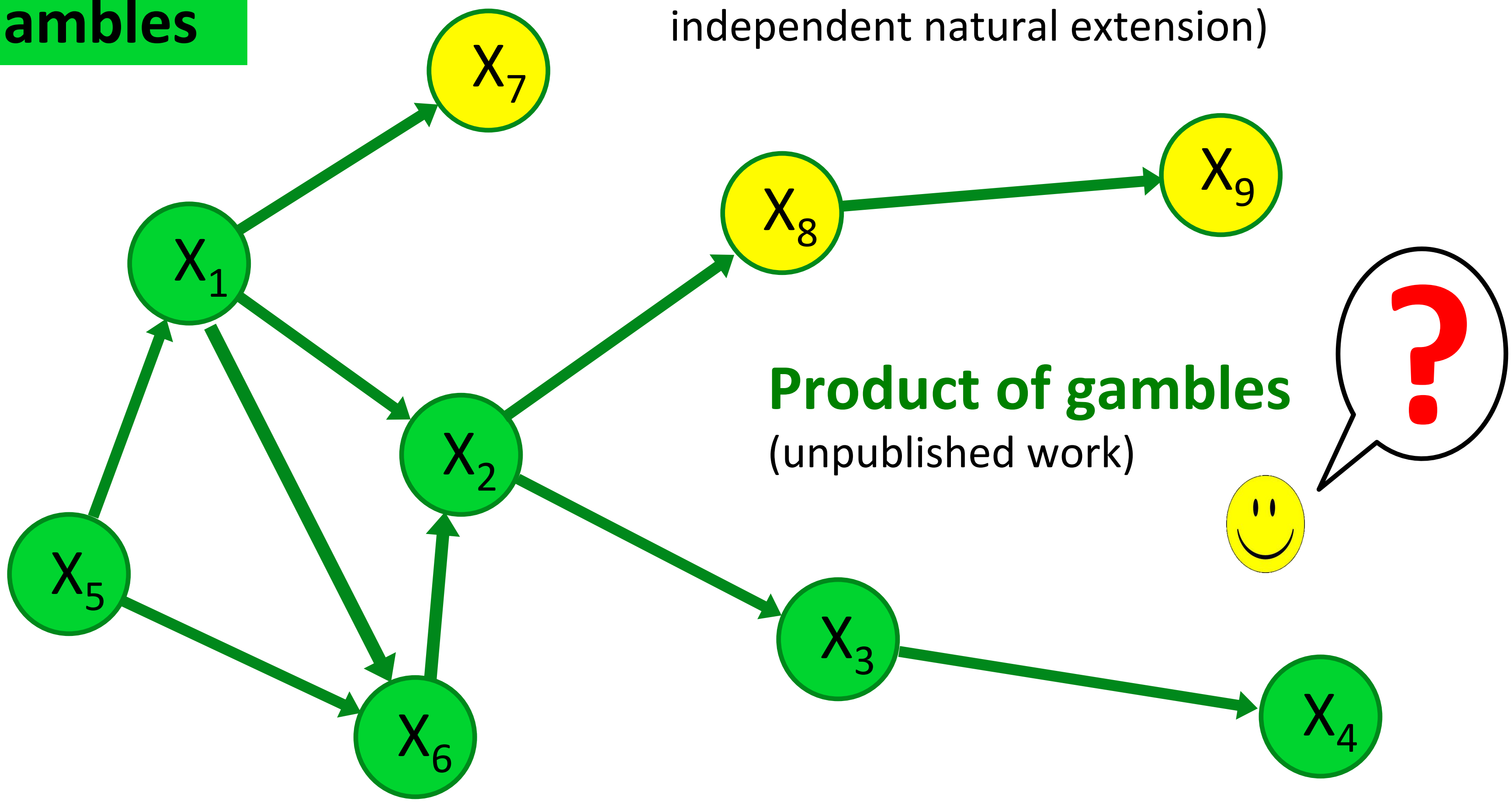
Example: time averages in queueing systems

efficient algorithms (linear for trees!)

Non-negative gambles

Gambles

(the joint can be constructed recursively by applying marginal extension and independent natural extension)



Product of gambles
(unpublished work)