Credal Networks

Fabio G. Cozman - University of Sao Paulo, Brazil
fgcozman@usp.br

July 17, 2012

Outline

1. Motivation: why are graph-based "languages" useful?
2. Background: basics on Bayesian networks.
3. Credal networks: basic definitions.
4. Credal networks: basic theory .
5. Credal networks: basic applications.

Basic fact: everything is basic...

Motivation

- Graphs offer a compact and expressive language to express scenarios...
  - ... with many independent modules, with interacting/hierachical pieces that display dependence/independence.
  - ... with simplifying assumptions concerning dependence.

Motivation

- Graphs offer a compact and expressive language to express scenarios...
  - ... with many independent modules, with interacting/hierachical pieces that display dependence/independence.
  - ... with simplifying assumptions concerning dependence.

- It is possible to exploit the structure of the graph to obtain...
  - ... insights about theoretical properties.
  - ... gain in computational operations.

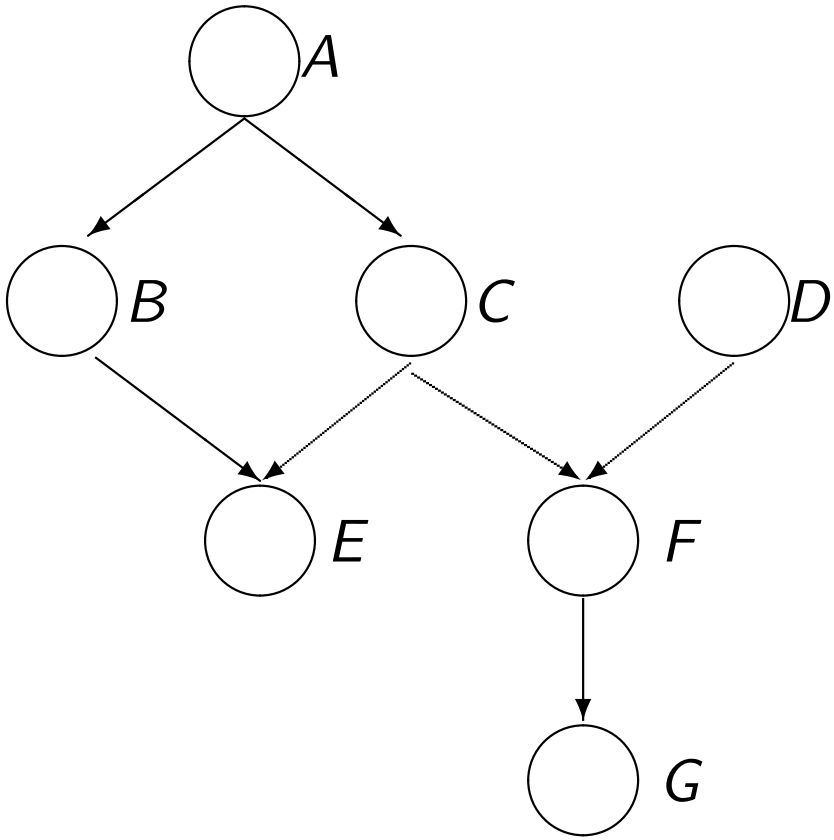
Motivation

- Graphs offer a compact and expressive language to express scenarios...
  - ... with many independent modules, with interacting/hierachical pieces that display dependence/independence.
  - ... with simplifying assumptions concerning dependence.

- It is possible to exploit the structure of the graph to obtain...
  - ... insights about theoretical properties.
  - ... gain in computational operations.

- So, graphs are great. Let' see what graphs are.

Detour: directed acyclic graph



Detour: graph-theoretic concepts

A	parent of	B and C
B and C	parents of	E
F	child of	C and D
E and F	children of	C
E, F and G	descendants of	C
A, B and D	nondescendants of	C
B, C, E, F and G	descendants of	A
D	nondescendant of	A
B and D	parents of children of	C

Motivation: Some history

- In the 60s, probabilities were not adopted in AI (McCarthy and Hayes: "information necessary to assign numerical probabilities is not ordinarily available')."

Motivation: Some history

- In the 60s, probabilities were not adopted in AI (McCarthy and Hayes: "information necessary to assign numerical probabilities is not ordinarily available')."
- Many alternatives to probability were adopted: certainty factors, Dempster-Shafer, fuzzy, non-classical logics...

Motivation: Some history

- In the 60s, probabilities were not adopted in AI (McCarthy and Hayes: "information necessary to assign numerical probabilities is not ordinarily available')."
- Many alternatives to probability were adopted: certainty factors, Dempster-Shafer, fuzzy, non-classical logics...
- During the 80s, probabilities received attention, and Bayesian networks appeared; Markov random fields were around and were adopted.

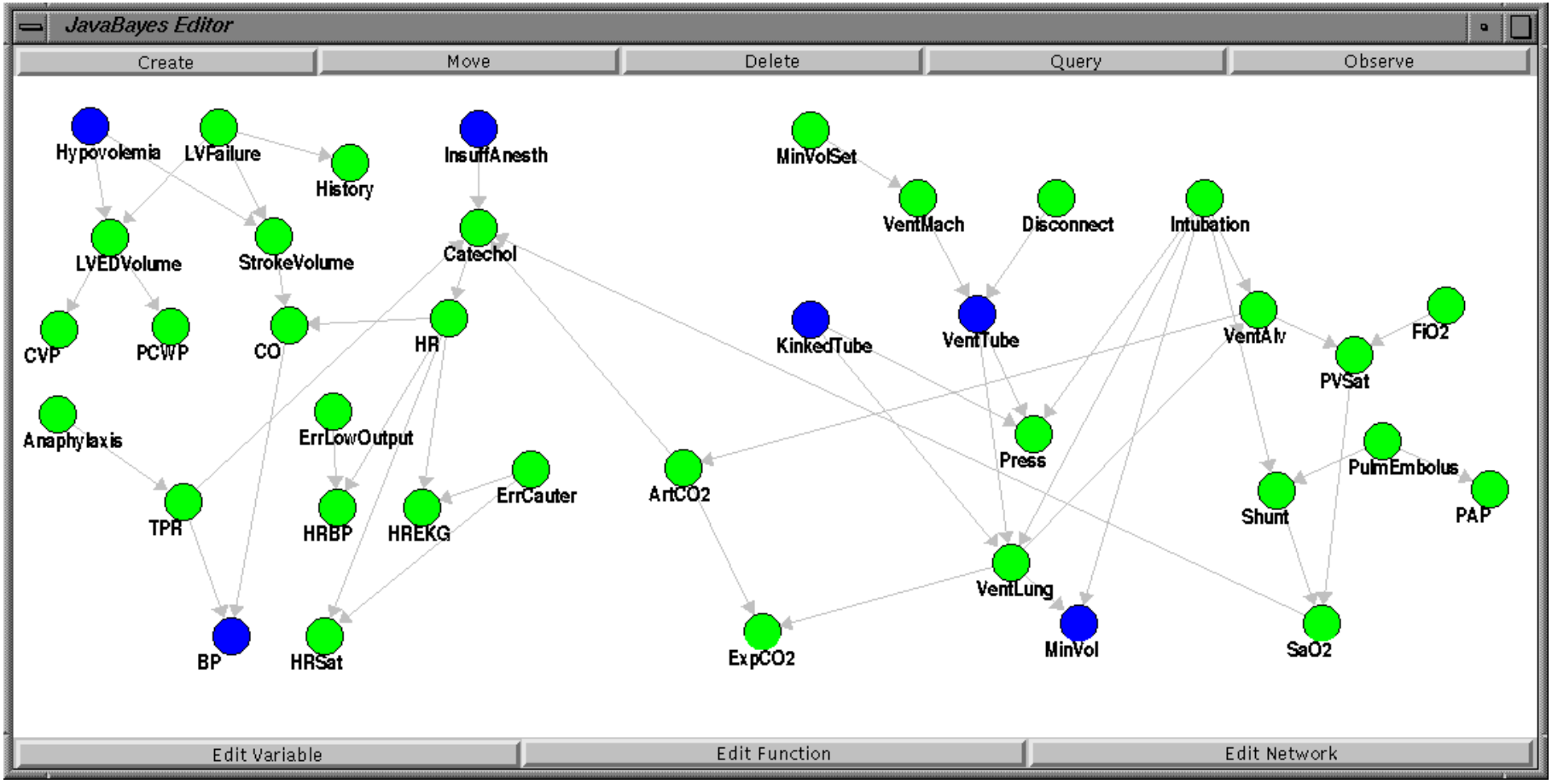
Motivation: Some history

- In the 60s, probabilities were not adopted in AI (McCarthy and Hayes: "information necessary to assign numerical probabilities is not ordinarily available')."
- Many alternatives to probability were adopted: certainty factors, Dempster-Shafer, fuzzy, non-classical logics...
- During the 80s, probabilities received attention, and Bayesian networks appeared; Markov random fields were around and were adopted.
- Since then, probability has been adopted everywhere: knowledge representation, planning and problem solving, learning.

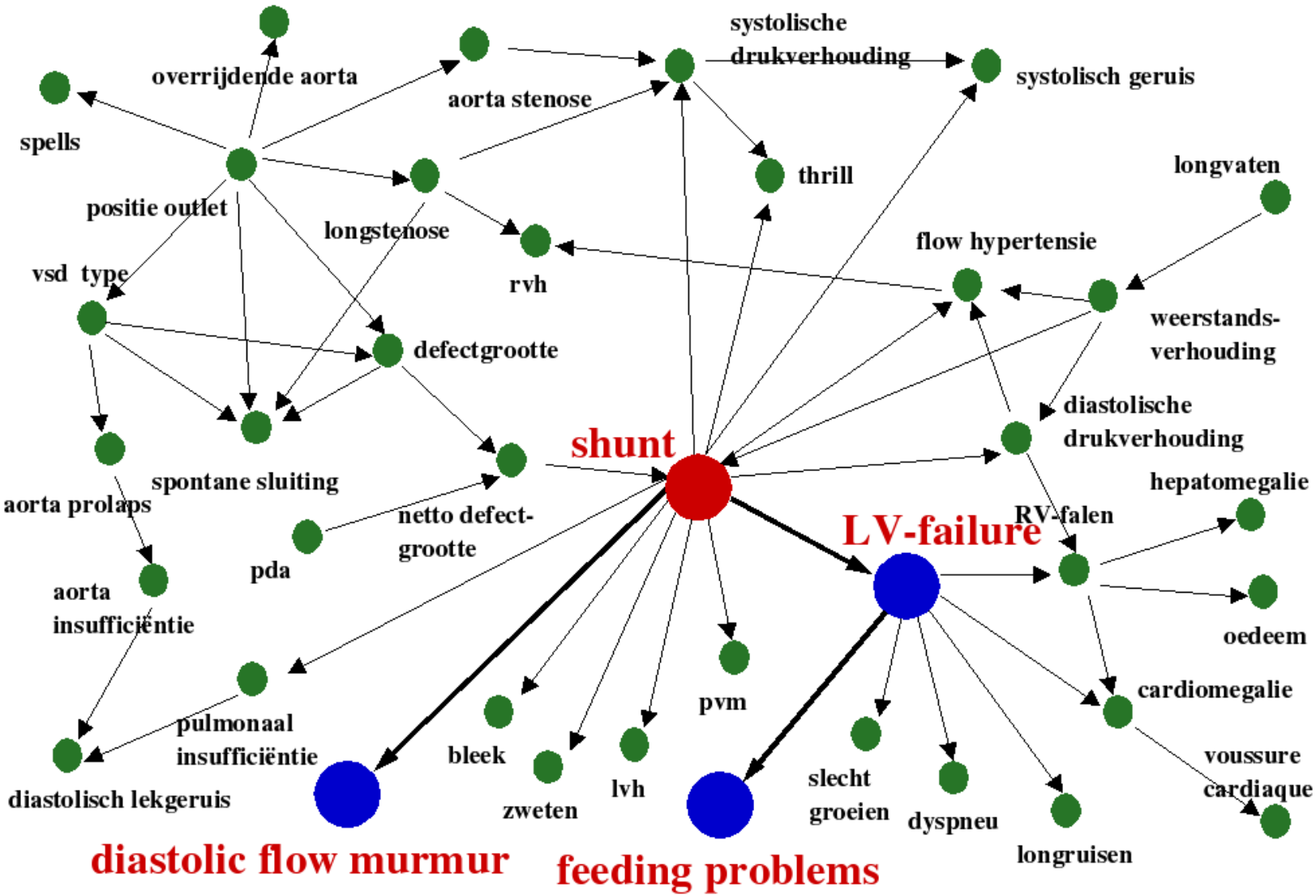
Motivation: Some history

- In the 60s, probabilities were not adopted in AI (McCarthy and Hayes: "information necessary to assign numerical probabilities is not ordinarily available')."
- Many alternatives to probability were adopted: certainty factors, Dempster-Shafer, fuzzy, non-classical logics...
- During the 80s, probabilities received attention, and Bayesian networks appeared; Markov random fields were around and were adopted.
- Since then, probability has been adopted everywhere: knowledge representation, planning and problem solving, learning.
- (Since 80s, credal networks have been also investigated.)

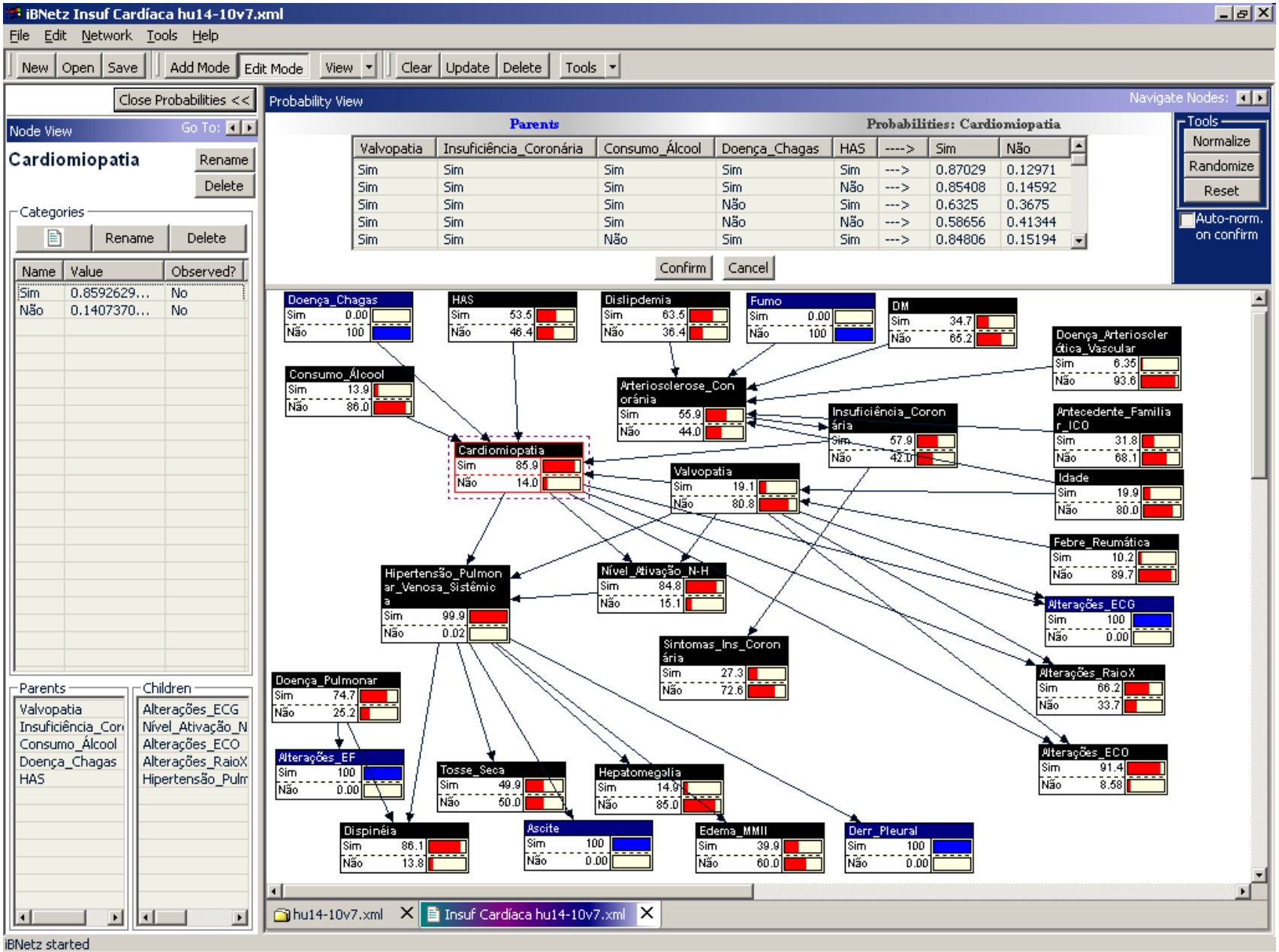
The Alarm network



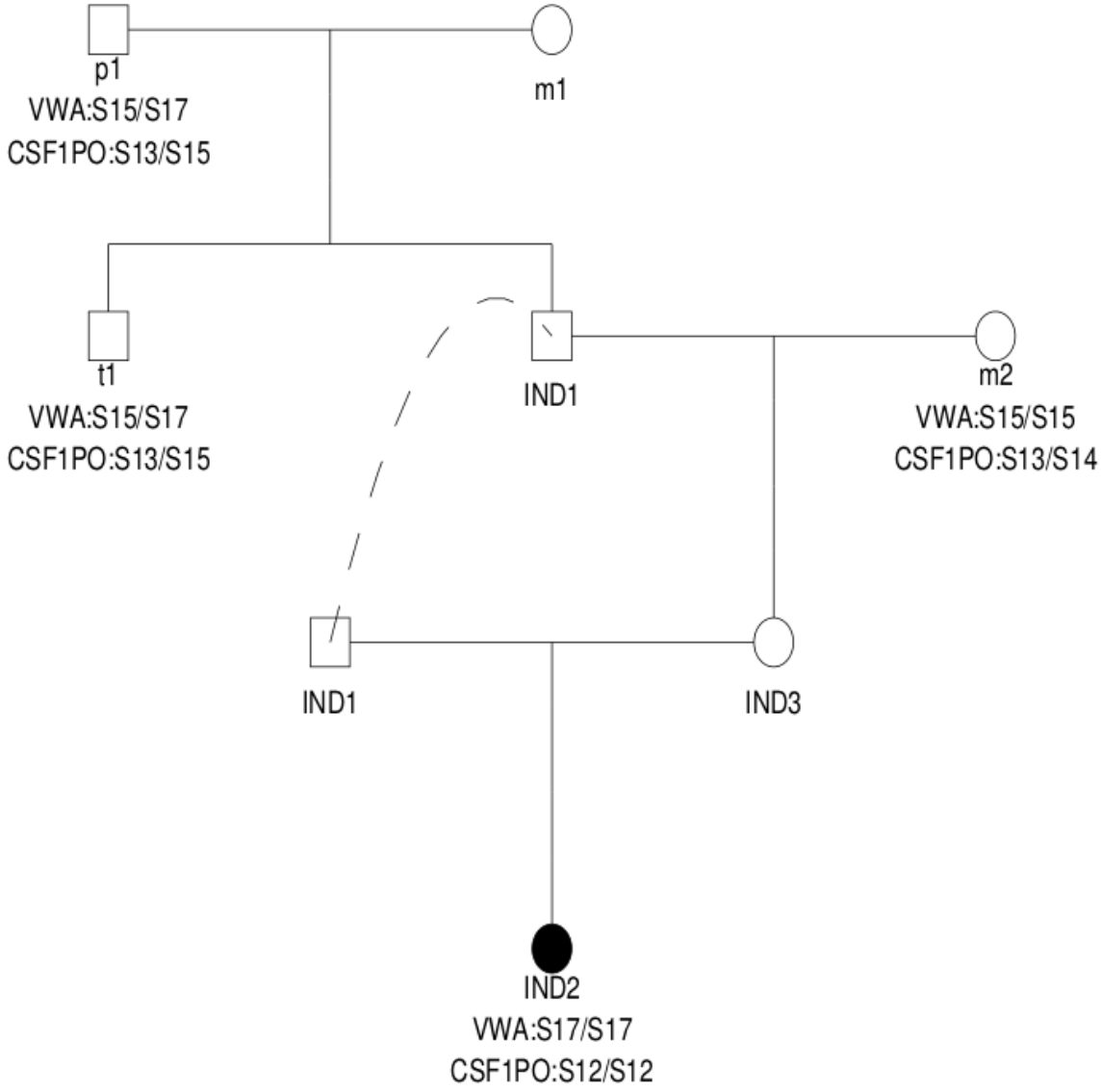
The LV-failure network



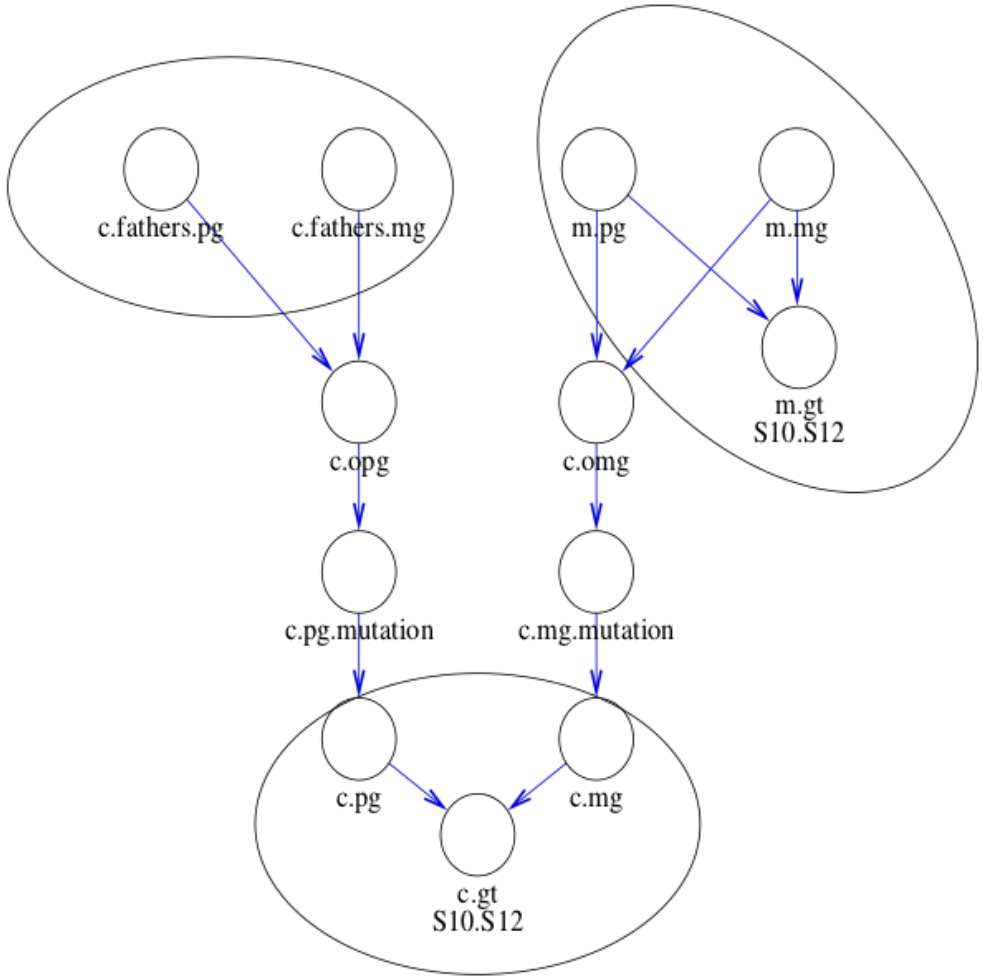
The HU network



Heredogram analysis

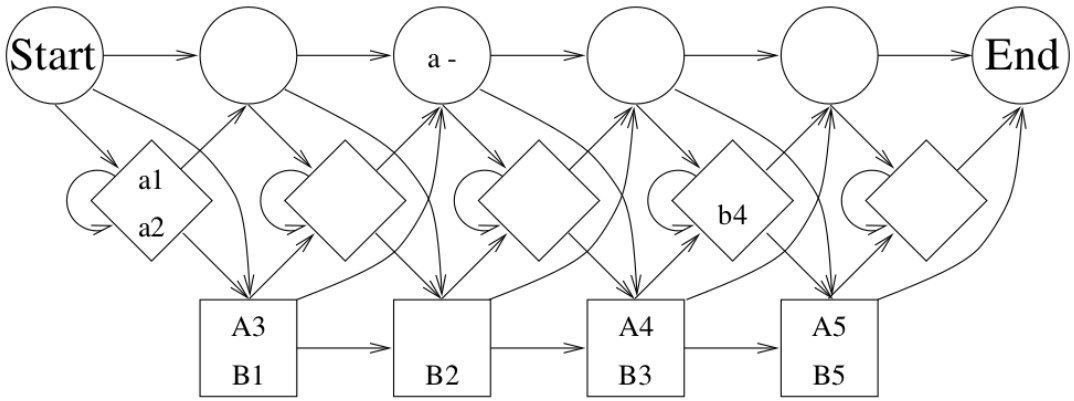


Heredogram to Bayesian network



Representing DNA sequences

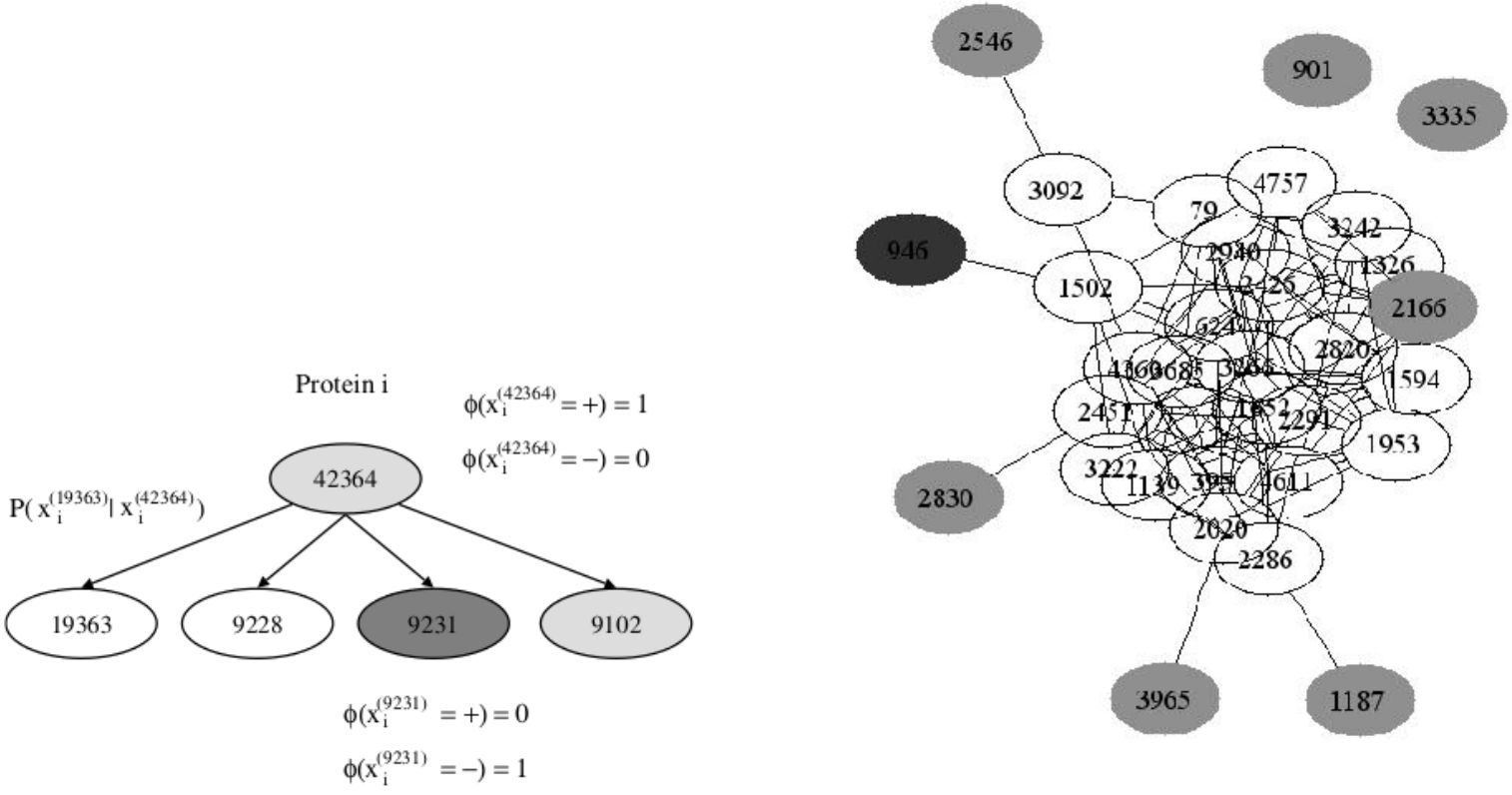
- A popular representation is based on Bayesian networks (actually, Hidden Markov Models):




```
a1 a2 A3     -     A4  .  A5
 .  .  B1    B2    B3 b4  B5
```

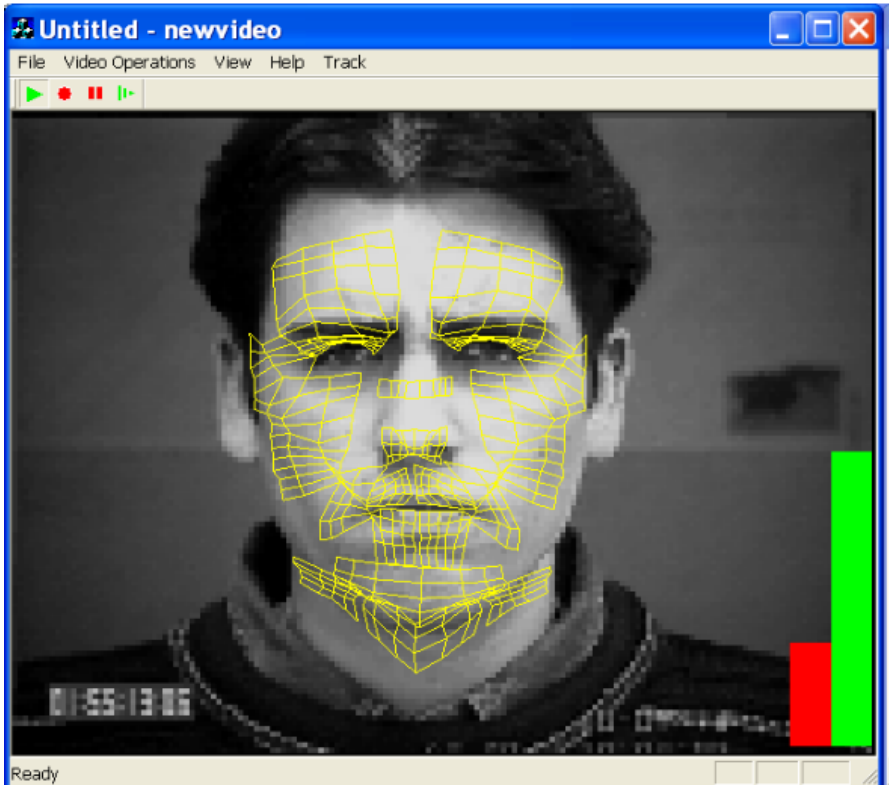
(from www.cse.ucsc.edu/compbio/sam.html)

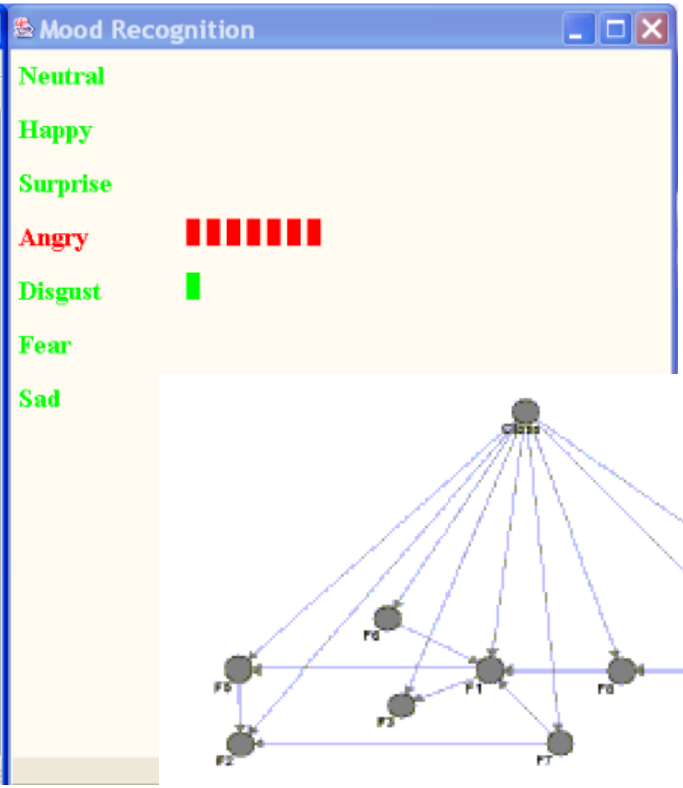
Protein interaction...

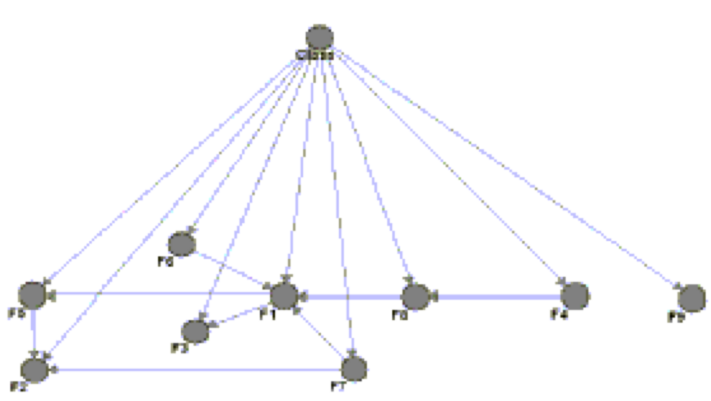


(Source: S. Carroll, V. Pavlovic, Protein classification using probabilistic chain graphs and the Gene Ontology structure. *Bioinformatics*, 22(15):1871–1878, 2006.)

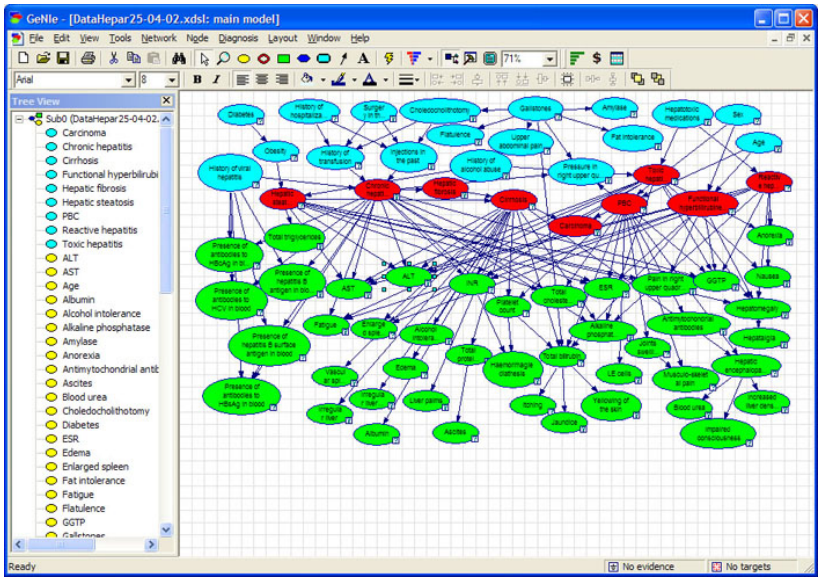
Classification: expression detection

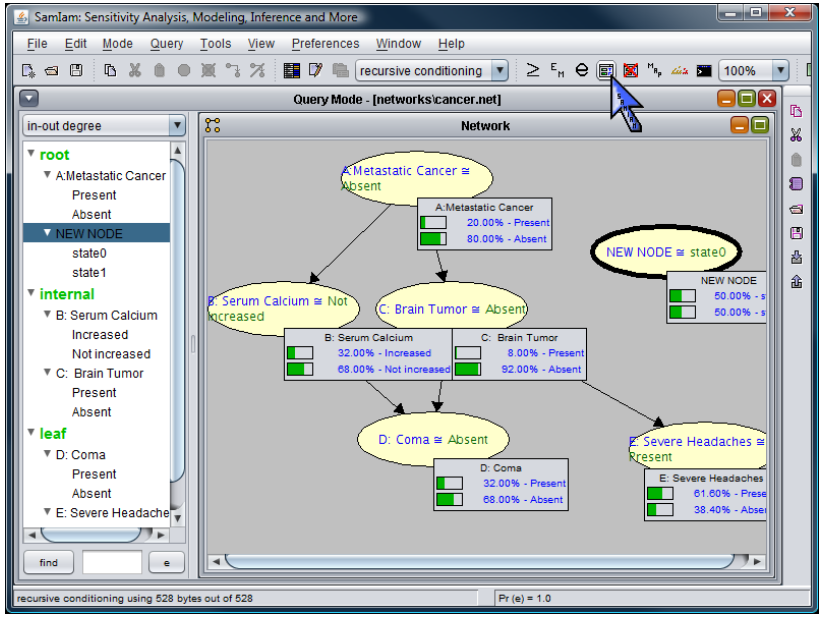






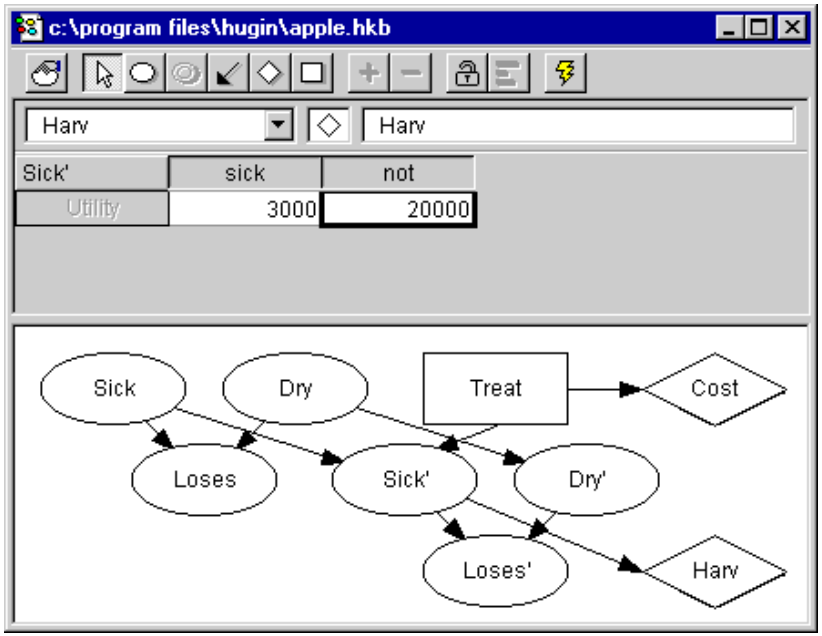
Genie, Samlam

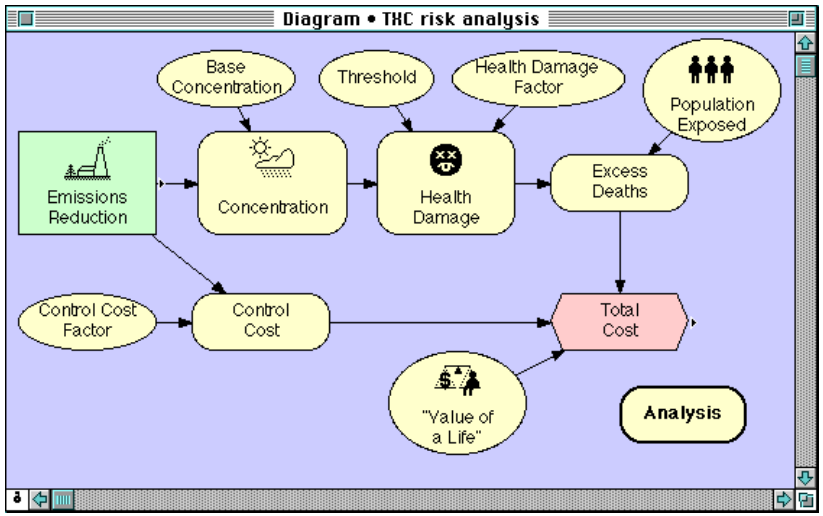




Others: BNT, and for more, bnt.sourceforge.net/bnsoft.html.

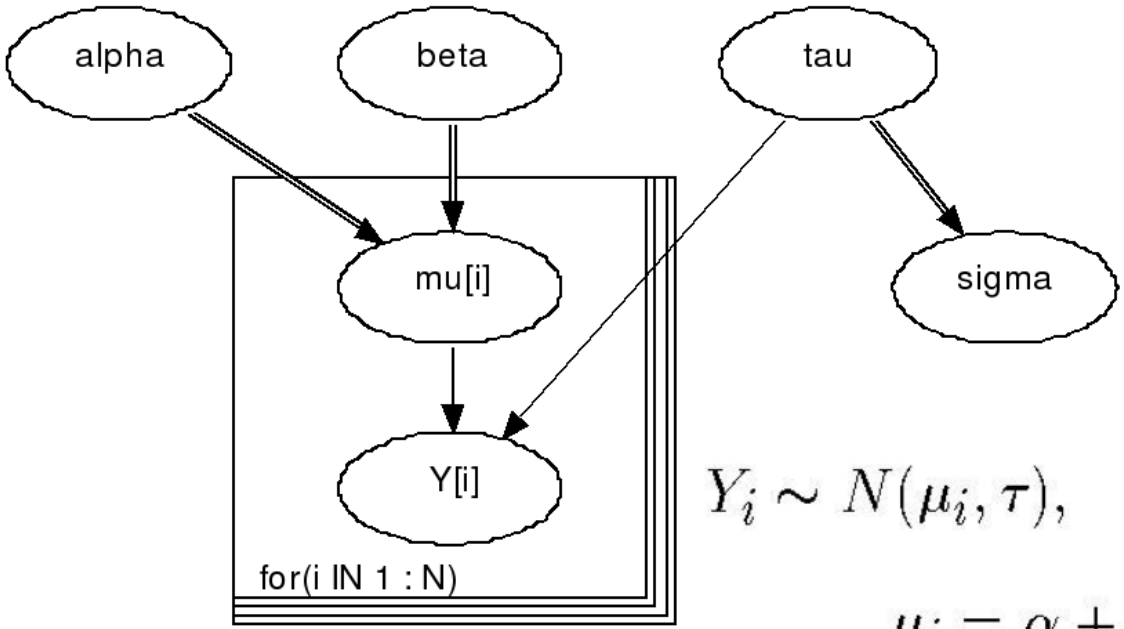
Hugin, Analytica





Others: Netica, BayesiaLab, and for more, bnt.sourceforge.net/bnsoft.html.

BUGS



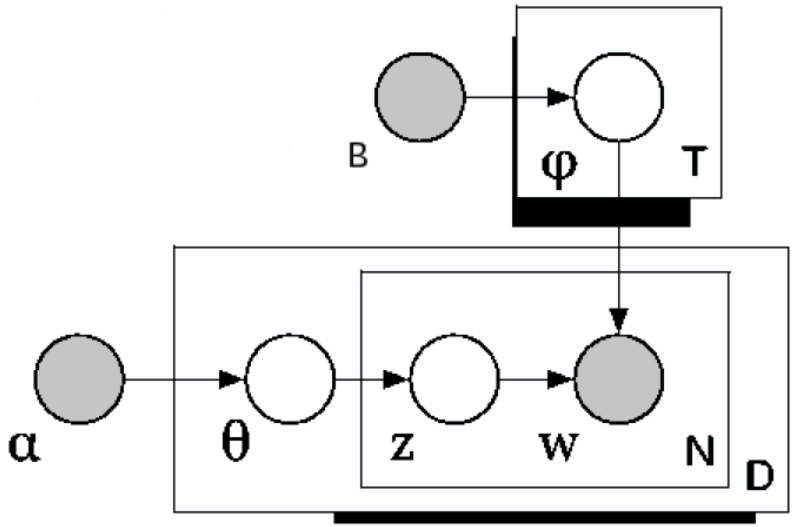
$$Y_i \sim N(\mu_i, \tau), \qquad i \in \{1, \ldots, N\}$$

$$\mu_i = \alpha + \beta(X_i - \bar{X})$$

This and a lot more: www.mrc-bsu.cam.ac.uk/bugs/

Application: Topic models

- Goal: to represent topics in text classification.
- Popular model: Latent Dirichlet analysis.



$$\varphi_j \sim \text{Dirichlet}(B), \quad \theta_d \sim \text{Dirichlet}(\alpha),$$

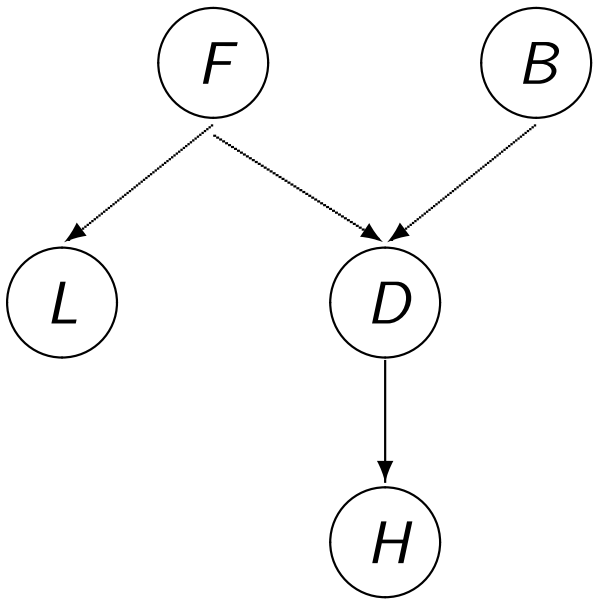
$$z_i \sim \text{Dirichlet}(\theta_{\beta_i}), \quad w_i | z_i = j \sim \text{Dirichlet}(\varphi_j).$$

Back to basics

- A Bayesian network encodes a single joint probability density over $\mathbf{X}$.
- The joint density is specified through a directed acyclic graph.
- Each node represents a random variable X_i in $\mathbf{X}$.
  - *Parents* of X_i: $\mathrm{pa}(X_i)$.

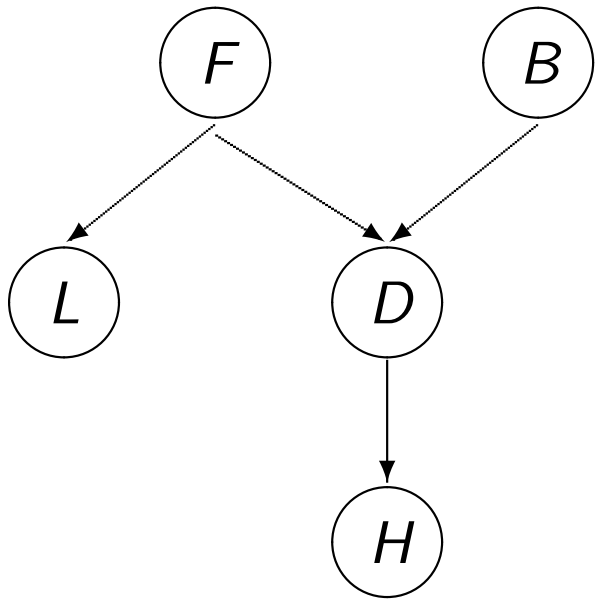
Example: The dog problem

By Charniak, 1991:



Semantics: The Markov condition

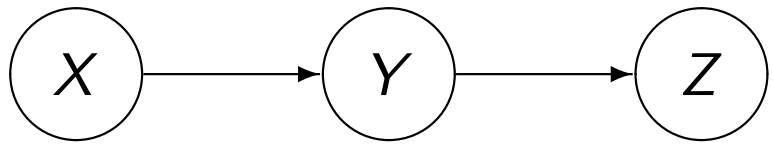
Every variable is independent of its nondescendants nonparents given its parents.



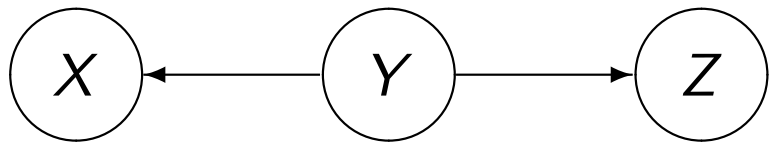
Exercise

Enumerate the independence relations implied by the Markov condition on these three networks.

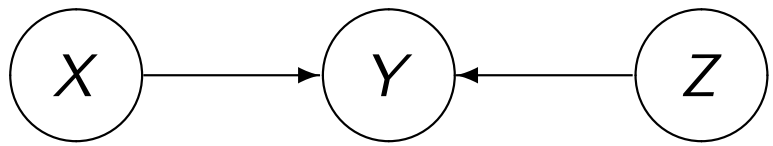
- CHAIN:



- FORK:



- COLLIDER:



Semantics: Basic result

Every variable is independent of its nondescendants nonparents given its parents.

Semantics: Basic result

Every variable is independent of its nondescendants nonparents given its parents.

Theorem

The Markov condition implies that any Bayesian network represents a unique joint probability density that factorizes as:

$$p(\mathbf{X}) = \prod_i p(X_i|\mathrm{pa}(X_i))\,.$$

Semantics: Basic result

Every variable is independent of its nondescendants nonparents given its parents.

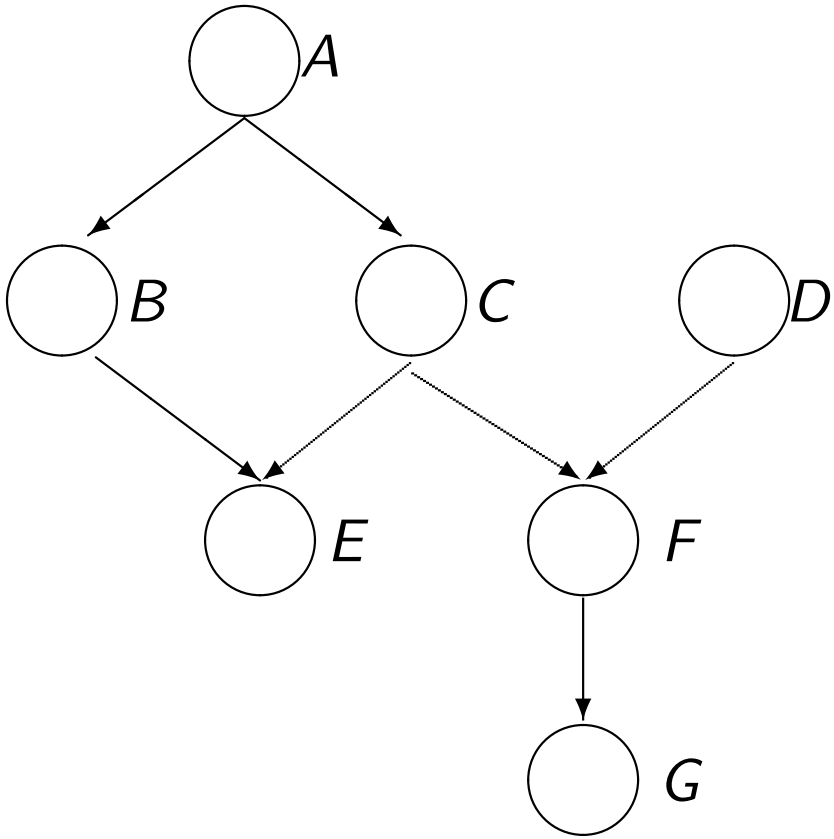
Theorem

The Markov condition implies that any Bayesian network represents a unique joint probability density that factorizes as:

$$p(\mathbf{X}) = \prod_i p(X_i|\mathrm{pa}(X_i))\,.$$

Such a factorization reduces the number of needed probability values.

Exercise



Write down the factorization of the joint distribution for $[A, B, C, D, E, F, G]$.

Exercise

1. Convince yourself that, given a directed acyclic graph, it is possible to order the variables in such a way that variable X_i is never preceded by one of its descendants.
2. Prove that for any Bayesian network,

$$p(\mathbf{X}) = \prod_i p(X_i|Y_i)\,,$$

   where Y_i is a set of variables that precede X_i in some ordering of variables that guarantees that X_i is never preceded by one of its descendants.
3. Now prove Theorem 1.

d-separation

- Famous concept in Bayesian networks.
- Very complicated; sound, but not complete.
- Conceptually important: allows one to discard pieces of the network.
- Proved only using the graphoid properties.
- Fast in a computer (polynomial algorithms).

d-separation

- Famous concept in Bayesian networks.
- Very complicated; sound, but not complete.
- Conceptually important: allows one to discard pieces of the network.
- Proved only using the graphoid properties.
- Fast in a computer (polynomial algorithms).

Definition

Given three sets of variables $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$, suppose that along every path between a variable in $\mathbf{X}$ and a variable in $\mathbf{Y}$ there is a variable W such that:

1. either W is a collider and is not in $\mathbf{Z}$ and none of its descendants are in $\mathbf{Z}$,
2. or W is not a collider and is in $\mathbf{Z}$.

Then $\mathbf{Y}$ and $\mathbf{X}$ are *d-separated* by $\mathbf{Z}$.

Perfect maps?

- A graph-based representation scheme where independence is equivalent to d-separation is a *perfect map*.

Perfect maps?

- A graph-based representation scheme where independence is equivalent to d-separation is a *perfect map*.
- Bayesian networks are not perfect maps: There may be independences without d-separation.

Perfect maps?

- A graph-based representation scheme where independence is equivalent to d-separation is a *perfect map*.
- Bayesian networks are not perfect maps: There may be independences without d-separation.
- Bayesian networks are *independence maps*: a d-separation implies an independence.

Perfect maps?

- A graph-based representation scheme where independence is equivalent to d-separation is a *perfect map*.
- Bayesian networks are not perfect maps: There may be independences without d-separation.
- Bayesian networks are *independence maps*: a d-separation implies an independence.
- Moreover, all independences implied by d-separation are obtained by application of graphoid properties to the Markov condition!

An exercise on failure of representation...

From Pearl, p. 126.

1. Consider population of animals where disease is spreading through sexual contact.
2. Closed heterosexual group: two males $M1$, $M2$ and two females $F1$, $F2$.
3. $M1$ and $M2$ are independent given $F1$ and $F2$.
4. $F1$ and $F2$ are independent given $M1$ and $M2$.
5. A pair of male-female is not independent.
6. Show: no Bayesian network with only four nodes can represent this.

Inference

- We want: $P(X_q|\mathbf{X}_E)$.

Inference

- We want: $P(X_q|\mathbf{X}_E)$.
- That is,

$$P(X_q|\mathbf{X}_E) = \frac{P(X_q, \mathbf{X}_E)}{P(\mathbf{X}_E)}$$

Inference

- We want: $P(X_q|\mathbf{X}_E)$.
- That is,

$$\begin{aligned} P(X_q|\mathbf{X}_E) &= \frac{P(X_q, \mathbf{X}_E)}{P(\mathbf{X}_E)} \\ &= \frac{\sum_{\mathbf{X}\setminus\{X_q,\mathbf{X}_E\}} P(\mathbf{X})}{\sum_{\mathbf{X}\setminus\mathbf{X}_E} P(\mathbf{X})} \end{aligned}$$

Inference

- We want: $P(X_q|\mathbf{X}_E)$.
- That is,

$$\begin{aligned} P(X_q|\mathbf{X}_E) &= \frac{P(X_q, \mathbf{X}_E)}{P(\mathbf{X}_E)} \\ &= \frac{\sum_{\mathbf{X}\setminus\{X_q,\mathbf{X}_E\}} P(\mathbf{X})}{\sum_{\mathbf{X}\setminus\mathbf{X}_E} P(\mathbf{X})} \\ &= \frac{\sum_{\mathbf{X}\setminus\{X_q,\mathbf{X}_E\}} \prod_i p(X_i|\mathrm{pa}(X_i))}{\sum_{\mathbf{X}\setminus\mathbf{X}_E} P(\prod_i p(X_i|\mathrm{pa}(X_i)))}. \end{aligned}$$

Inference

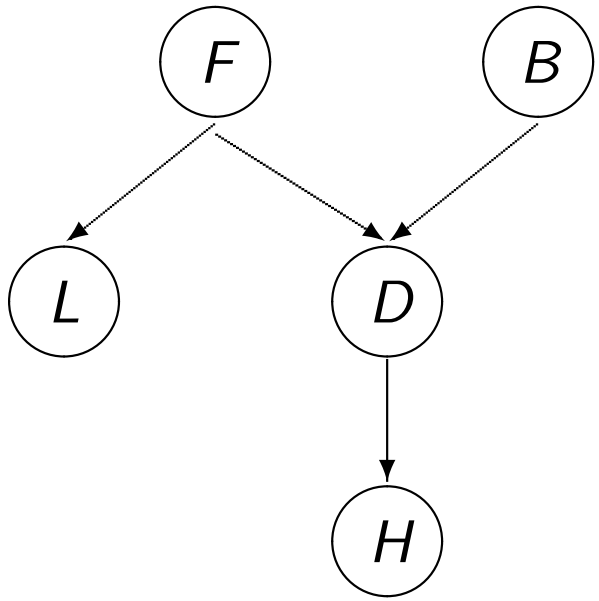
- We want: $P(X_q|\mathbf{X}_E)$.
- That is,

$$\begin{aligned} P(X_q|\mathbf{X}_E) &= \frac{P(X_q, \mathbf{X}_E)}{P(\mathbf{X}_E)} \\ &= \frac{\sum_{\mathbf{X}\setminus\{X_q,\mathbf{X}_E\}} P(\mathbf{X})}{\sum_{\mathbf{X}\setminus\mathbf{X}_E} P(\mathbf{X})} \\ &= \frac{\sum_{\mathbf{X}\setminus\{X_q,\mathbf{X}_E\}} \prod_i p(X_i|\mathrm{pa}(X_i))}{\sum_{\mathbf{X}\setminus\mathbf{X}_E} P(\prod_i p(X_i|\mathrm{pa}(X_i)))}. \end{aligned}$$

- Problem is #P-hard; some special cases are easy (polytrees: Pearl algorithm), and some algorithms work well in practice...
  - There are very powerful algorithms to approximate this (MCMC, variational, loopy).
- There are other inferences... to be mentioned later.

Exercise

Compute $P(F|L)$.



$$p(f) = 0.5 \qquad p(b) = 0.5$$
$$p(l|f) = 0.6 \qquad p(l|f^c) = 0.05$$
$$p(d|f,b) = 0.8 \qquad p(d|f,b^c) = 0.1$$
$$p(d|f^c,b) = 0.1 \qquad p(d|f^c,b^c) = 0.7$$
$$p(h|d) = 0.6 \qquad p(h|d^c) = 0.3$$

Learning

- Get *training* data, produce graph/probability values.
- Maximum likelihood: counting, and perhaps the EM algorithm.
- Bayesian: typically with conjugate priors and independence assumptions; often through MCMC or variational approximations.

Building a Bayesian network

- Elicitation from experts.
  - Start identifying variables, and build the graph.
  - Then elicit the numbers.
- Learning from data.
  - Elicit graph, learn numbers.
  - Learn graph and numbers.
- ... or any combination of expert opinion and data.

In short,

1. Bayesian networks are compact and intuitive.
2. They consist of graph and conditional distributions.
3. Basic assumption is Markov condition.

In short,

1. Bayesian networks are compact and intuitive.
2. They consist of graph and conditional distributions.
3. Basic assumption is Markov condition.
4. The method is not "perfect" in a technical sense (not all independences can be represented).
   - Should we try other kinds of graphs? We could: Markov random fields, chain graphs, etc. None is "perfect"... Anyway, not discussed in this talk...

Credal networks

- Suppose we have a set of variables $\mathbf{X}$.
- We wish to *compactly* specify a *set* of joint distributions over $\mathbf{X}$, using graphs.
- Maybe we just wish to specify a set of Bayesian networks.
- Or, maybe we wish to specify a set of joint distributions using a single graph and associated credal sets.
  - This is a credal network.

Defining credal networks

- Take a directed acyclic graph, with a variable associated with each node.

Defining credal networks

- Take a directed acyclic graph, with a variable associated with each node.

- Two possibilities:
  1. We assume that every node is associated with a "local" credal set/lower prevision conditional on its parents, and some rule that combines these local pieces.
     - Maybe *impose* $p(\mathbf{X}) = \prod_i p(X_i|\mathrm{pa}(X_i))$?
  2. We assume a Markov condition of some sort, and see what happens.
     - Maybe *derive* $p(\mathbf{X}) = \prod_i p(X_i|\mathrm{pa}(X_i))$?

Defining strong extension

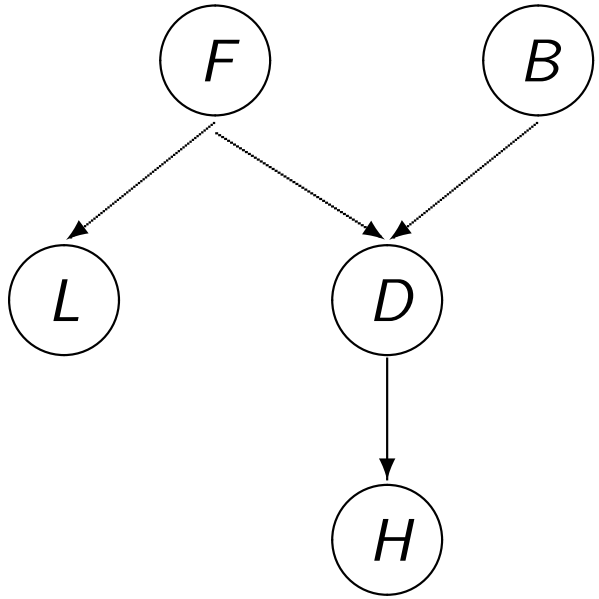
- Suppose we have a directed acyclic graph, with a variable associated with each node.
- Suppose every node is strongly independent of its nondescendants nonparents given its parents.
- Take the largest set of joint distributions that satisfies this condition (the *strong* extension).
- What is this set?

Strong extension

- The strong extension is the convex hull of a set of joint distributions, all of which factorize as

$$p(\mathbf{X}) = \prod_i p(X_i|\mathrm{pa}(X_i))\,.$$

Example: The credal dog problem



$$p(f) \in [0.5, 0.6] \qquad p(b) \leq 0.5$$
$$p(l|f) \in [0.5, 0.7] \qquad p(l|f^c) = [0.1, 0.2]$$
$$p(d|f, b) = 0.8 \qquad p(d|f, b^c) = 0.1$$
$$p(d|f^c, b) = 0.1 \qquad p(d|f^c, b^c) = 0.7$$
$$p(h|d) \in [0.5, 0.8] \qquad p(h|d^c) = 0.3$$

Note: in the previous example,

- there are 2^5 possible ways to factorize the joint distribution... Number of vertices is the main challenge with strong extensions.

- all local credal sets are *separately specified*; we might instead have a constraint $p(I|f) + p(I|f^c) \geq 0.5$. Or maybe even a *non-local* constraint $p(I|f) \geq p(h|d)$...

Separately specified strong extension

- The strong extension is the convex hull of a set of joint distributions, all of which factorize as

$$p(\mathbf{X}) = \prod_i p(X_i|\mathrm{pa}(X_i))\,.$$

- Hence each variable can be associated with a "local" credal sets $K(X_i|\mathrm{pa}(X_i) = \pi_{ik})$, one for each valid π_{ik}.

Separately specified strong extension

- The strong extension is the convex hull of a set of joint distributions, all of which factorize as

$$p(\mathbf{X}) = \prod_i p(X_i|\mathrm{pa}(X_i))\,.$$

- Hence each variable can be associated with a "local" credal sets $K(X_i|\mathrm{pa}(X_i) = \pi_{ik})$, one for each valid π_{ik}.
- Vertices of the joint credal set are combinations of vertices of the "local" credal sets.
- Any lower/upper expectation is attained at a vertex of the joint credal set (thus at a combination of vertices of "local" credal sets.

Exercise

- One can imagine a set of joint distributions that satisfies the "strong" Markov condition but that is smaller than the strong extension... just imagine imposing constraints amongst the local distribution!

- Challenge: construct such a set of joint distributions for a network $X \to Y$, where X and Y are binary variables.

Specifying local conditional credal sets

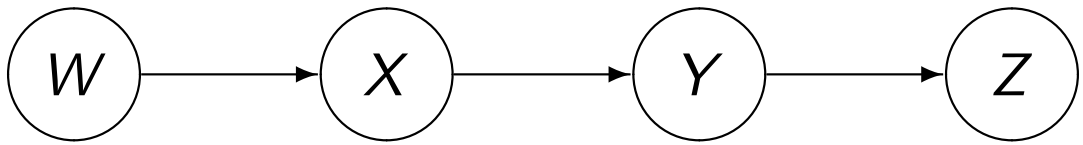
- Qualitative constraints (back to Wellman, 1990).
- Probability intervals (back at least to Tessem, 1992).
- Order of magnitude comparisons.
- Belief functions.
- Possibilistic measures.
- General constraints (back a long way, van der Gaag, Moral,...), either on probabilities or on lower probabilities/expectations.

Defining epistemic extension

- Suppose we have a directed acyclic graph, with a variable associated with each node.
- Suppose every node is epistemically independent of its nondescendants nonparents given its parents.
- Take the largest set of joint distributions that satisfies this condition (the *epistemic* extension).
- What is this set?

An example of epistemic extension

Consider network



with four binary variables, and probability intervals assigned to all probability values.
More than 6 million vertices in the epistemic extension!!!
For more on epistemic extension, see Gert' talk.

Epistemic independence and d-separation

- Epistemic independence does not satisfy the contraction property.
- A credal network with epistemic independence may not satisfy d-separation.
- Example:
  - Binary variables W, X and Y.
  - $K(W, X, Y)$ is convex hull of three distributions:

W	X	Y	$p_1(X, Y, W)$	$p_2(X, Y, W)$	$p_3(X, Y, W)$
W_0	X_0	Y_0	0.008	0.018	0.0093
W_1	X_0	Y_0	0.072	0.072	0.0757
W_0	X_1	Y_0	0.032	0.042	0.037
W_1	X_1	Y_0	0.288	0.168	0.228
W_0	X_0	Y_1	0.096	0.084	0.09
W_1	X_0	Y_1	0.024	0.126	0.075
W_0	X_1	Y_1	0.384	0.196	0.290
W_1	X_1	Y_1	0.096	0.294	0.195

  - X and Y are epistemically independent; X and W are conditionally epistemically independent given Y.
  - But X and (W, Y) are not not epistemically independent.

Epistemic independence and a conjecture

- Epistemic independence does not satisfy the contraction property.
- A credal network with epistemic independence may not satisfy d-separation.
  - Perhaps the way to go is to pursue different graph-based models (Moral, Vantaggi).
  - Perhaps the way to go is to assume just epistemic irrelevance.
- Conjecture: the epistemic extension does satisfy d-separation.

Inference with strong extensions

- Strong extensions are quite similar to Bayesian networks (for instance, d-separation).
- Inference is:

$$\underline{P}(X_q|\mathbf{X}_E) = \min P(X_q|\mathbf{X}_E)\,.$$

Inference with strong extensions

- Strong extensions are quite similar to Bayesian networks (for instance, d-separation).
- Inference is:

$$\underline{P}(X_q|\mathbf{X}_E) = \min P(X_q|\mathbf{X}_E)\,.$$

- Or, more explicitly,

$$\min \frac{\sum_{\mathbf{X}\setminus\{X_q,\mathbf{X}_E\}} \prod_i p(X_i|\mathrm{pa}(X_i))}{\sum_{\mathbf{X}\setminus\mathbf{X}_E} \prod_i p(X_i|\mathrm{pa}(X_i))}.$$

where typically the min is over a large set of "local" credal sets.

Inference with strong extensions

- Strong extensions are quite similar to Bayesian networks (for instance, d-separation).
- Inference is:

$$\underline{P}(X_q|\mathbf{X}_E) = \min P(X_q|\mathbf{X}_E)\,.$$

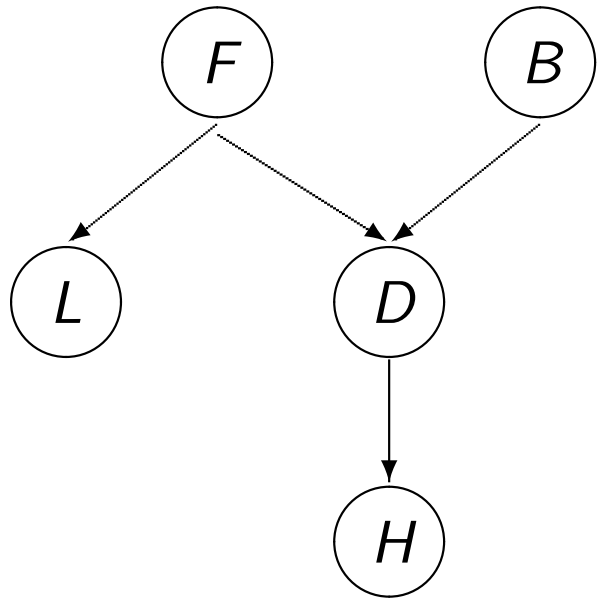
- Or, more explicitly,

$$\min \frac{\sum_{\mathbf{X}\setminus\{X_q,\mathbf{X}_E\}} \prod_i p(X_i|\mathrm{pa}(X_i))}{\sum_{\mathbf{X}\setminus\mathbf{X}_E} \prod_i p(X_i|\mathrm{pa}(X_i))}.$$

  where typically the min is over a large set of "local" credal sets.
- This is a multilinear program.
- Solution lies at a set of vertices of credal sets.
- Best solutions resort to optimization theory to procedure inference engines.

Exercise

Compute $\underline{P}(F|L)$.



$$p(f) \in [0.5, 0.6] \qquad p(b) \leq 0.5$$
$$p(l|f) \in [0.5, 0.7] \qquad p(l|f^c) = [0.1, 0.2]$$
$$p(d|f, b) = 0.8 \qquad p(d|f, b^c) = 0.1$$
$$p(d|f^c, b) = 0.1 \qquad p(d|f^c, b^c) = 0.7$$
$$p(h|d) \in [0.5, 0.8] \qquad p(h|d^c) = 0.3$$

Inference methods

- Enumeration methods (obsolete), multilinear programming (exact/approximate), simulated annealing, genetic algorithms... (approximate), variational (approximate).

- Special case where exact inference is simple: polytrees with binary variables.
  - Polytrees: if one discards arrow directions, one gets a tree.
  - Binary variables: credal sets are intervals.

2U and loopy 2U

- In the polytree+binary variable case, the 2U algorithm produces inferences in polynomial time.
- The 2U algorithm can be understood as a sequence of message exchanges between nodes in the polytree; the sequence surely ends.
- The *loopy* 2U algorithm is an approximate method for general graphs: messages are continuously exchanged, until probability intervals for all nodes are obtained (no guarantees, but good practical performance).

Complexity

Bayesian Networks

Problem	*Polytree*	*Bounded induced-width*	*General*
Belief updating	Polynomial	Polynomial	PP-Complete
MPE	Polynomial	Polynomial	NP-Complete
MAP	NP-Complete	NP-Complete	NP^{PP}-Complete
MmAP	Σ_2^{P}-Complete	Σ_2^{P}-Complete	NP^{PP}-Hard

Strong extensions

Problem	*Polytree*	*Bounded induced-width*	*General*
Belief updating	NP-Complete	NP-Complete	NP^{PP}-Complete
MPE	Polynomial	Polynomial	NP-Complete
MAP	Σ_2^{P}-Complete	Σ_2^{P}-Complete	NP^{PP}-Hard

(Bounded induced-width: subjacent graph has induced-width bounded by O(log(s)), where s is the size of input.)

Learning

- First scenario: missing data.
  Absence of assumptions concerning missing data leads to set of estimates for probability values.

- Example:
  Consider network $X \rightarrow Y$ and data:

X	0	1	0	1	1
Y	1	0	0	1	?

  Here, maximum likelihood estimate for $P(Y = 1|X = 1)$ belongs to $[1/3, 2/3]$.

Learning with imprecise priors

- Second scenario: imprecise priors. For instance, instead of Dirichlet distributions, the Imprecise Dirichlet model (IDM).

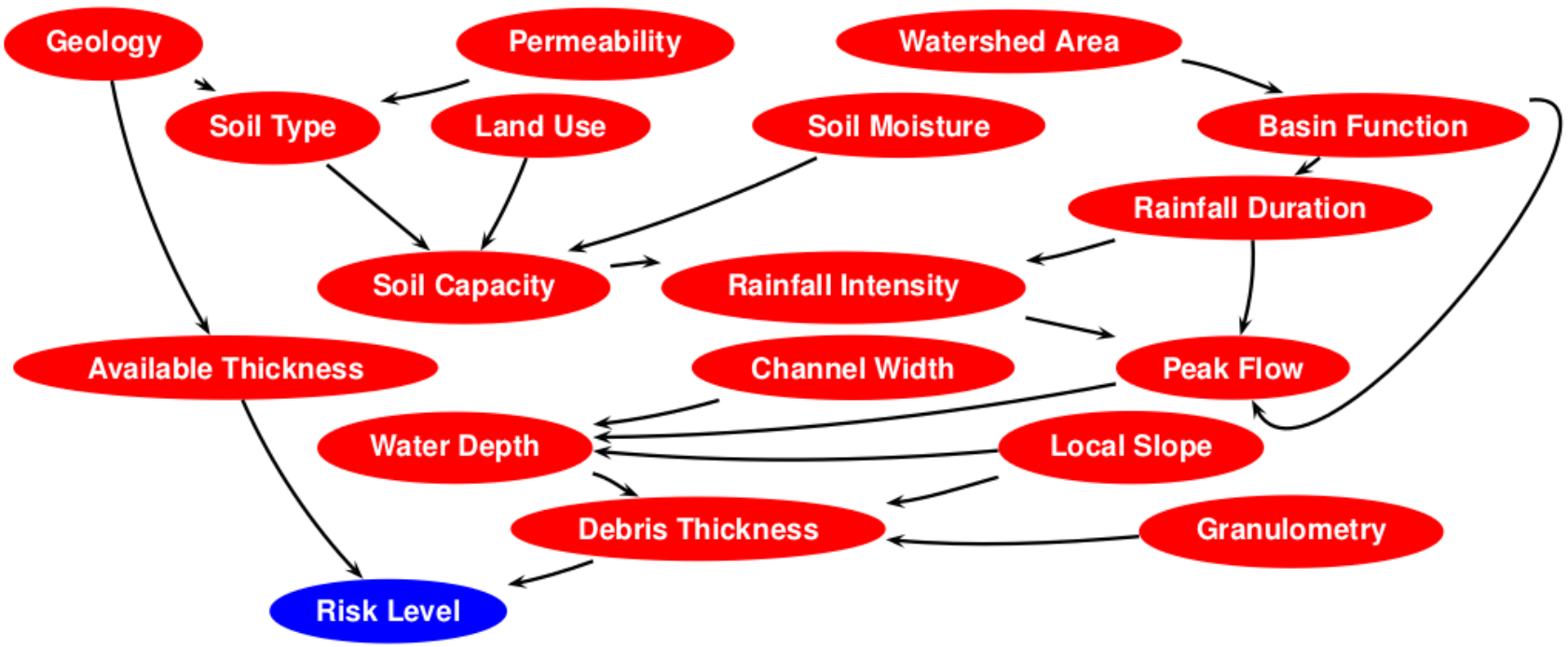
- Here we have, with suitable independence assumptions over the priors:

$$\hat{P}(X_i = x_{ij} | \mathrm{pa}(X_i) = \pi_{ik}) \in \left[\frac{n_{ijk}}{s + n_{ik}}, \frac{s + n_{ijk}}{s + n_{ik}} \right],$$

  where s is the parameter of the IDM.

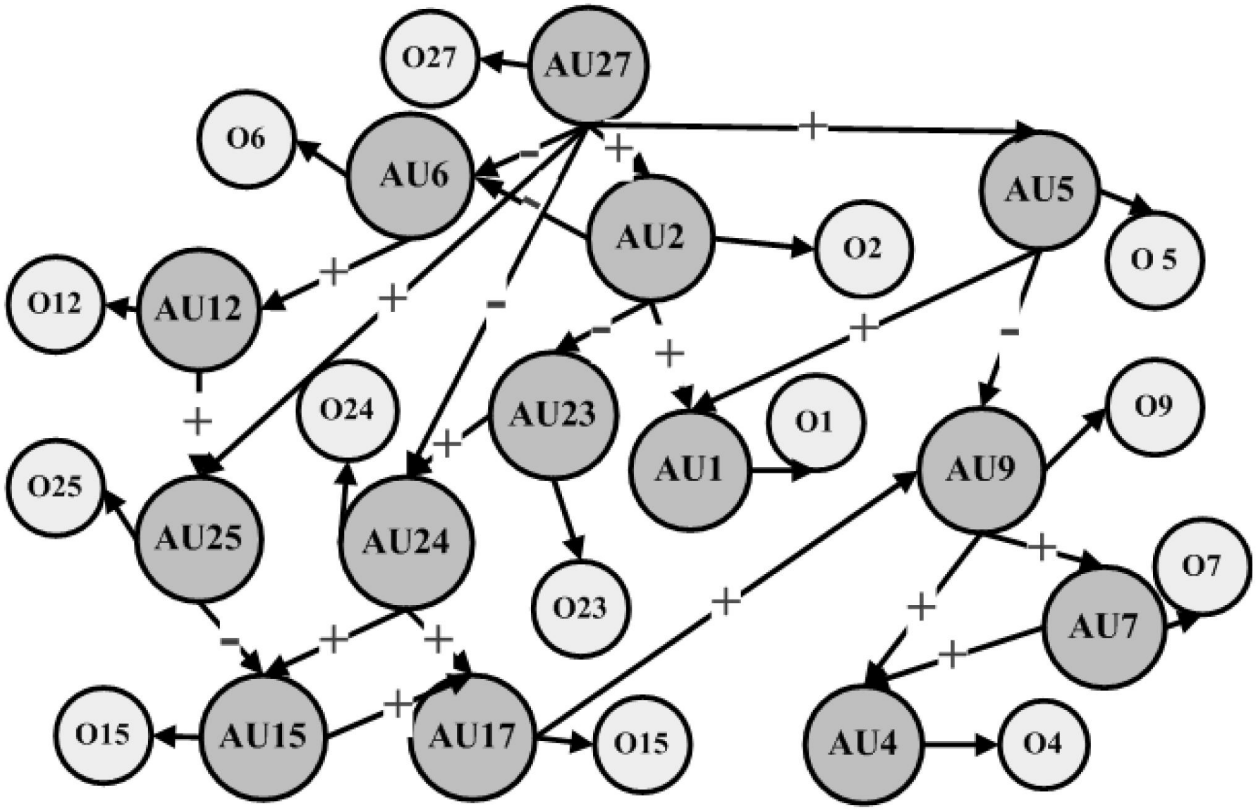
Application I

Expert system for assessment of debris in Switzerland (IDSIA).



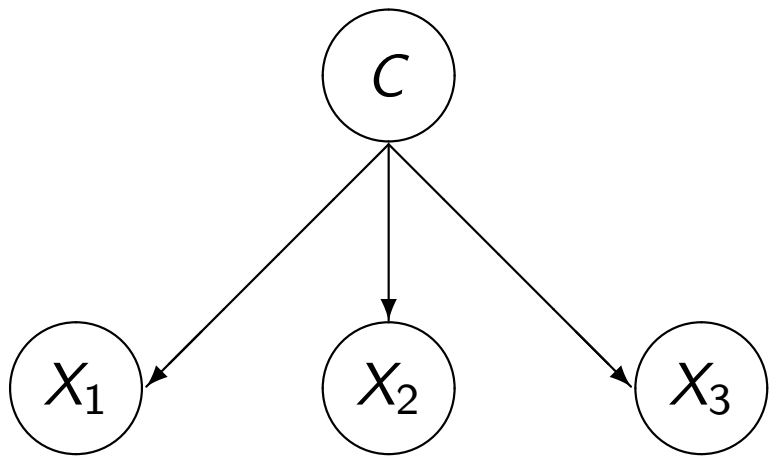
Application II

Semi-qualitative prior for facial expression recognition (IDSIA).



Application III

Credal classifiers: dealing with missing values, few data points, and imprecise priors in classification (IDSIA).



JNCC2: www.idsia.ch/~giorgio/jncc2.html

Application III

Credal classifiers: dealing with missing values, few data points, and imprecise priors in classification (IDSIA).

Data set	*NBC*	*NCC2-MAR*				*NCC2-nonMAR*				*Subsets of instances*				
	Acc. (%)	Det. (%)	Single-Acc. (%)	SetAcc. (%)	Indet.OutSize	Det. (%)	Single-Acc. (%)	SetAcc. (%)	Indet.OutSize	NCC2-MAR D (%)	NCC2-MAR I (%)	NCC2-nonMAR D (%)	NCC2-nonMAR I (%)	ΔNCC2-nonMAR (%)
ecoli (8 cl.)	85.0	84.2	88.5	92.2	3.8	54.4	95.0	96.6	5.2	88.5	66.6 (6.4)	95.0	72.9 (3.4)	74.7 (4.3)
glass (6 cl.)	67.5	52.1	80.4	79.5	2.4	22.4	81.9	94.8	3.7	80.4	53.6 (3.9)	81.9	63.3 (2.6)	61.6 (4.9)
haberman (2 cl.)	72.0	91.8	74.2	100.0	2.0	66.7	80.8	100.0	2.0	74.2	47 (10.9)	80.8	54.1 (4.8)	56.5 (6.2)
kr-kp (2 cl.)	86.8	98.4	87.4	100.0	2.0	5.4	100.0	100.0	2.0	87.4	48 (10.2)	100.0	86.1 (0.4)	86.7 (0.4)
letter (26 cl.)	72.4	91.6	77.0	57.7	2.7	13.3	99.0	97.9	18.7	77.0	22.0 (1.2)	99.0	68.3 (0.2)	68.4 (0.2)
monks1 (2 cl.)	70.0	87.4	72.7	100.0	2.0	39.7	82.5	100.0	2.0	72.7	51.2 (5.8)	82.5	61.6 (3.0)	64.3 (3.6)
monks2 (2 cl.)	62.2	86.6	63.4	100.0	2.0	15.3	81.2	100.0	2.0	63.4	54.6 (6.8)	81.2	58.9 (2.2)	59.7 (2.3)
monks3 (2 cl.)	95.3	99.2	95.5	100.0	2.0	67.3	96.7	100.0	2.0	95.6	64 (25)	96.6	91.7 (3.2)	92.5 (3.1)
nursery (5 cl.)	87.0	97.5	87.4	78.9	2.0	49.0	98.4	99.6	2.5	87.4	66.1 (5.3)	98.4	76.0 (1.0)	76.0 (0.9)
optdigits (10 cl.)	91.2	95.2	93.5	86.9	2.7	14.8	99.9	99.9	8.0	93.5	46.9 (3.6)	99.9	89.7 (0.2)	89.8 (0.2)
pendigits (10 cl.)	87.2	96.3	89.5	81.8	2.5	29.2	97.1	99.1	7.2	89.5	26.7 (2.7)	97.1	83.1 (0.3)	83.2 (0.3)
segment (7 cl.)	91.0	89.3	95.6	96.7	3.7	31.7	98.8	99.9	6.1	95.6	52.5 (3.9)	98.8	87.4 (0.7)	88.7 (0.7)
sonar (2 cl.)	84.4	90.9	86.9	100.0	2.0	41.8	97.9	100.0	2.0	86.9	59 (14.3)	97.9	74.6 (3.1)	77.4 (2.6)
spambase (2 cl.)	89.1	99.5	89.4	100.0	2.0	42.1	97.2	100.0	2.0	89.4	31 (13.4)	97.2	83.3 (0.5)	83.7 (0.4)
spect (2 cl.)	76.1	90.8	79.3	100.0	2.0	53.9	90.2	100.0	2.0	79.3	44 (12.6)	90.2	59.8 (4.1)	63.7 (4.3)
splice (3 cl.)	94.5	97.2	96.4	96.2	2.2	0.0	100.0	100.0	3.0	96.4	49.5 (7.6)	100.0	94.9 (0.2)	94.5 (0.2)
waveform (3 cl.)	81.3	99.0	81.6	99.9	2.0	19.7	94.5	100.0	2.3	81.6	50.3 (7.6)	94.5	78.0 (0.4)	78.2 (0.2)
yeast (10 cl.)	56.6	91.9	58.6	70.0	2.3	28.0	69.1	91.1	3.7	58.5	34.9 (6.7)	69.1	51.7 (1.7)	51.7 (1.7)

Data set	*NBC*	*NCC2-MAR*				*NCC2-nonMAR*				*Subsets of instances*				
	Acc. (%)	Det. (%)	Single-Acc. (%)	SetAcc. (%)	Indet.OutSize	Det. (%)	Single-Acc. (%)	SetAcc. (%)	Indet.OutSize	NCC2-MAR D (%)	NCC2-MAR I (%)	NCC2-nonMAR D (%)	NCC2-nonMAR I (%)	ΔNCC2-nonMAR (%)
ecoli (8cl.)	85.2	83.2	88.9	92.6	3.7	69.6	94.9	94.1	3.8	88.9	67.2 (5.7)	94.9	62.9 (3.0)	60.4 (6.7)
glass (6 cl.)	67.5	47.5	81.7	79.7	2.4	29.6	83.5	91.1	3.0	81.7	54.8 (3.4)	83.5	60.7 (2.2)	58.7 (6.9)
haberman (2 cl.)	72.0	93.5	73.8	100.0	2.0	79.4	78.5	100.0	2.0	73.8	47 (12.2)	78.5	46.6 (5.7)	47.1 (9.6)
kr-kp (2 cl.)	88.0	98.5	88.6	100.0	2.0	52.9	96.8	100.0	2.0	88.6	49.1 (9.4)	96.8	78.1 (1.0)	79.0 (0.9)
letter (26 cl.)	72.9	91.9	77.4	58.5	2.8	35.3	96.7	91.8	11.7	77.4	22.0 (1.1)	96.7	59.9 (0.3)	60.0 (0.3)
monks1 (2 cl.)	71.0	87.5	73.7	100.0	2.0	58.8	80.1	100.0	2.0	73.7	51.8 (5.5)	80.1	57.9 (2.4)	60.6 (3.2)
monks2 (2 cl.)	62.2	86.7	63.5	100.0	2.0	40.2	72.5	100.0	2.0	63.5	53.5 (6.0)	72.5	55.6 (2.2)	56.3 (2.5)
monks3 (2 cl.)	95.5	99.8	95.6	100.0	2.0	90.5	97.2	100.0	2.0	96.3	64 (31.3)	87.6	73 (14.1)	73 (14.1)
nursery (5 cl.)	87.8	97.5	88.3	80.3	2.0	68.8	97.2	99.6	2.3	88.3	67.8 (4.5)	97.2	67.1 (1.5)	66.9 (1.2)
optdigits (10 cl.)	91.3	95.3	93.5	86.2	2.6	54.6	99.0	98.4	4.9	93.5	45.2 (3.1)	99.0	82.0 (0.4)	82.4 (0.4)
pendigits (10 cl.)	87.4	96.3	89.7	81.8	2.5	57.1	95.5	96.8	5.4	89.7	25.7 (2.5)	95.5	76.5 (0.5)	77.3 (0.5)
segment (7 cl.)	91.6	89.5	95.9	97.6	3.7	54.8	97.7	99.3	5.5	95.9	54.9 (3.0)	97.7	84.2 (0.9)	87.8 (1.0)
sonar (2 cl.)	83.8	91.8	87.1	100.0	2.0	56.5	94.4	100.0	2.0	87.1	46 (17)	94.4	69.9 (3.2)	75.2 (3.2)
spambase (2 cl.)	88.9	99.5	89.2	100.0	2.0	76.1	94.1	100.0	2.0	89.2	39 (12.8)	94.1	72.6 (0.8)	73.3 (0.8)
spect (2 cl.)	73.4	88.5	78.1	100.0	2.0	67.4	83.5	100.0	2.0	78.1	36.5 (8.8)	83.5	52.3 (4.0)	60.9 (4.7)
splice (3 cl.)	95.1	97.6	96.8	96.3	2.2	0.1	100.0	100.0	2.8	96.2	52.1 (5.5)	100.0	95.1 (0.2)	95.0 (0.2)
waveform (3 cl.)	81.4	99.1	81.6	99.8	2.0	54.1	89.6	100.0	2.1	81.6	49.2 (5.5)	89.6	71.7 (0.6)	72.0 (0.6)
yeast (10 cl.)	57.2	91.5	59.1	72.1	2.3	55.1	67.3	87.5	2.9	59.0	36.7 (6.7)	67.3	44.8 (1.8)	45.1 (1.9)

JNCC2: www.idsia.ch/˜giorgio/jncc2.html

Application IV

CR$\mathcal{ALC}$: description logic with probabilities, applied to mobile robotics (USP).

$$\begin{aligned} \mathsf{Desk} \equiv \mathsf{Table} \sqcap \exists \mathsf{near}.\mathsf{Chair}, \quad & \mathsf{Entrance} \equiv \mathsf{Door} \sqcap \exists \mathsf{near}.\mathsf{Sign}, \\ \mathsf{InteriorObject} \quad & \sqsubseteq \quad \mathsf{Object}, \\ P(\mathsf{Object}) \in [0.2, 0.8]\,, \quad & P(\mathsf{Environment}) \in [0.2, 0.8]\,. \end{aligned}$$

Application V

Planning with Markov Decision Processes with Imprecise Probabilities: graph-based representations of transition credal sets (USP).

```
(define (domain sysadmin)
(:requirements :adl)(:types comp)(:predicates (up ?c)(conn ?c ?d))
(:action reboot
  :parameters (?x - comp)
  :effect
     (and (decrease (reward) 1)
          (probabilistic 0.9 (up ?x))
          (oneof
          (forall (?d - comp)
            (probabilistic
              0.6 (when (exists (?c - comp)
                     (and (conn ?c ?d)(not (up ?c))(not (= ?x ?d))))
                  (not (up ?d))
          )))
          (forall (?d - comp)
            (probabilistic
              0.8 (when (exists (?c - comp)
                     (and (conn ?c ?d)(not (up ?c))(not (= ?x ?d))))
                  (not (up ?d))
     )))))))
```

To conclude...

- Graph-based "languages" can be used to compactly encode probabilities and credal sets over several variables.
- Credal networks are quite flexible and expressive.
- There are several possible extensions for a credal network.
  - Strong extension is the most popular.
  - Epistemic extension is quite intuitive.
- Inference and learning methods have been developed.
- Applications have been addressed.