CSC384: Lecture 11

- Last time
  - D-separation and inference in belief networks
- Today
  - Variable elimination; decision making; utility theory
  - Readings:
    - Today: 10.3, 10.4 (utility theory)
    - Next week: 10.4 (decision trees and decision nets)
- Announcements:
  - Check out BN construction/evaluation applet on CI Web page:
  - **http://www.cs.ubc.ca/labs/lci/CIspace/bayes**

Variable Elimination

- The intuitions in examples from last time give us a simple inference algorithm for networks without loops: the *polytree* algorithm. We won't discuss it further. But be comfortable with the intuitions.
- Instead we'll look at a more general algorithm that works for general BNs; but the propagation algorithm will more or less be a special case.
- The algorithm, ***variable elimination***, simply applies the summing out rule repeatedly. But to keep computation simple, it exploits the independence in the network and the ability to distribute sums inward.

Factors

- A function $f(X_1, X_2, \ldots, X_k)$ is also called a *factor*. We can view this as table of numbers, one for each instantiation of the variables $X_1, X_2, \ldots, X_k$.
- A tabular rep'n of a factor is exponential in k
- Each CPT in a Bayes net is a factor:
  - e.g., Pr(C|A,B) is a function of three variables, A, B, C
- Notation: f(**X**,**Y**) denotes a factor over the variables **X** ∪ **Y**. (Here **X**, **Y** are *sets* of variables.)

The Product of Two Factors

- Let f(**X**,**Y**) & g(**Y**,**Z**) be two factors with variables **Y** in common
- The *product* of f and g, denoted h = f x g (or sometimes just h = fg), is defined:

$$h(\mathbf{X},\mathbf{Y},\mathbf{Z}) = f(\mathbf{X},\mathbf{Y}) \times g(\mathbf{Y},\mathbf{Z})$$

f(A,B)		g(B,C)		h(A,B,C)			
ab	0.9	bc	0.7	abc	0.63	ab~c	0.27
a~b	0.1	b~c	0.3	a~bc	0.08	a~b~c	0.02
~ab	0.4	~bc	0.8	~abc	0.28	~ab~c	0.12
~a~b	0.6	~b~c	0.2	~a~bc	0.48	~a~b~c	0.12

Summing a Variable Out of a Factor

- Let $f(X,\mathbf{Y})$ be a factor with variable X ($\mathbf{Y}$ is a set)
- We *sum out* variable X from f to produce a new factor $h = \Sigma_X f$, which is defined:

$$h(\mathbf{Y}) = \Sigma_{x \in Dom(X)} f(x,\mathbf{Y})$$

f(A,B)		h(B)	
ab	0.9	b	1.3
a~b	0.1	~b	0.7
~ab	0.4		
~a~b	0.6		

Restricting a Factor

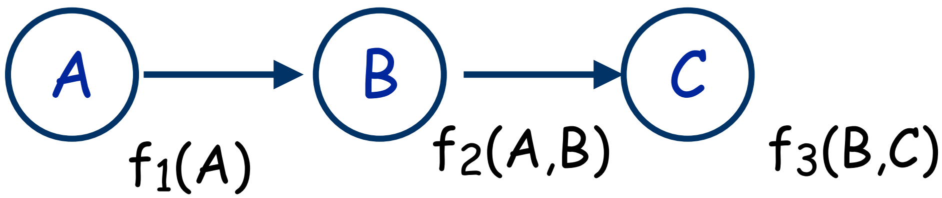
- Let f(X,**Y**) be a factor with variable X (**Y** is a set)
- We *restrict* factor f *to* X=x by setting X to the value x and "deleting". Define $h = f_{X=x}$ as:

$$h(\mathbf{Y}) = f(x,\mathbf{Y})$$

f(A,B)		$h(B) = f_{A=a}$	
ab	0.9	b	0.9
a~b	0.1	~b	0.1
~ab	0.4		
~a~b	0.6		

Variable Elimination: No Evidence

- Computing prior probability of query var X can be seen as applying these operations on factors

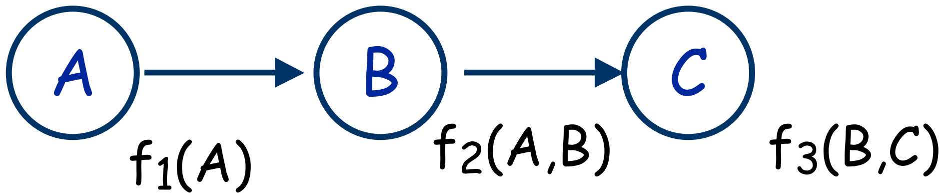


$$
\begin{aligned}
P(C) &= \Sigma_{A,B}\, P(C|B)\, P(B|A)\, P(A) \\
&= \Sigma_{B}\, P(C|B)\, \Sigma_{A}\, P(B|A)\, P(A) \\
&= \Sigma_{B}\, f_3(B,C)\, \Sigma_{A}\, f_2(A,B)\, f_1(A) \\
&= \Sigma_{B}\, f_3(B,C)\, f_4(B) \\
&= f_5(C)
\end{aligned}
$$

Define new factors: $f_4(B) = \Sigma_A\, f_2(A,B)\, f_1(A)$ and $f_5(C) = \Sigma_B\, f_3(B,C)\, f_4(B)$

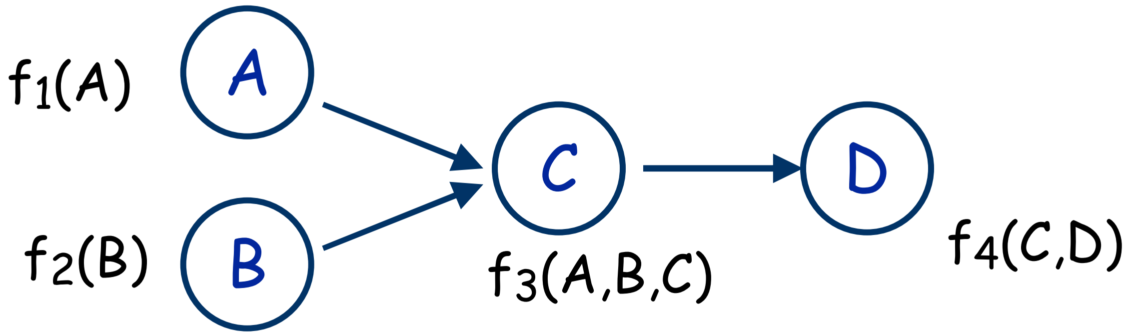
Variable Elimination: No Evidence

- Here's the example with some numbers



$f_1(A)$		$f_2(A,B)$		$f_3(B,C)$		$f_4(B)$		$f_5(C)$	
a	0.9	ab	0.9	bc	0.7	b	0.85	c	0.625
~a	0.1	a~b	0.1	b~c	0.3	~b	0.15	~c	0.375
		~ab	0.4	~bc	0.2				
		~a~b	0.6	~b~c	0.8				

VE: No Evidence (Example 2)



$$\begin{aligned}
P(D) &= \Sigma_{A,B,C}\ P(D|C)\ P(C|B,A)\ P(B)\ P(A) \\
&= \Sigma_C\ P(D|C)\ \Sigma_B\ P(B)\ \Sigma_A\ P(C|B,A)\ P(A) \\
&= \Sigma_C\ f_4(C,D)\ \Sigma_B\ f_2(B)\ \Sigma_A\ f_3(A,B,C)\ f_1(A) \\
&= \Sigma_C\ f_4(C,D)\ \Sigma_B\ f_2(B)\ f_5(B,C) \\
&= \Sigma_C\ f_4(C,D)\ f_6(C) \\
&= f_7(D)
\end{aligned}$$

Define new factors: $f_5(B,C)$, $f_6(C)$, $f_7(D)$, in the obvious way

Variable Elimination: One View

- One way to think of variable elimination:
  - write out desired computation using the chain rule, exploiting the independence relations in the network
  - arrange the terms in a convenient fashion
  - distribute each sum (over each variable) in as far as it will go
    - i.e., the sum over variable X can be "pushed in" as far as the "first" factor mentioning X
  - apply operations "inside out", repeatedly eliminating and creating new factors (note that each step/removal of a sum eliminates one variable)

Variable Elimination Algorithm

- Given query var Q, remaining vars **Z**. Let F be set of factors corresponding to CPTs for {Q} ∪ **Z**.

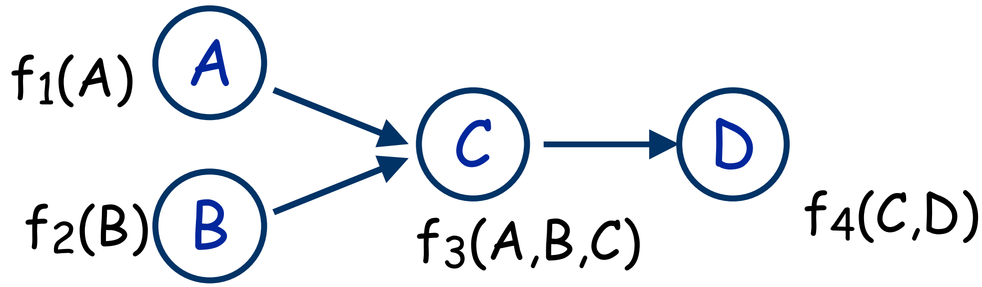
1. Choose an elimination ordering $Z_1, \ldots, Z_n$ of variables in **Z**.
2. For each Z_j -- in the order given -- eliminate $Z_j \in \mathbf{Z}$ as follows:
   (a) Compute new factor $g_j = \Sigma_{Zj} f_1 \times f_2 \times \ldots \times f_k$,
   where the f_i are the factors in F that include Z_j
   (b) Remove the factors f_i (that mention Z_j) from F and add new factor g_j to F
3. The remaining factors refer only to the query variable Q. Take their product and normalize to produce P(Q)

VE: Example 2 again

Factors: $f_1(A)$ $f_2(B)$ $f_3(A,B,C)$ $f_4(C,D)$

Query: P(D)?

Elim. Order: A, B, C



Step 1: Add $f_5(B,C) = \Sigma_A f_3(A,B,C)\, f_1(A)$

Remove: $f_1(A)$, $f_3(A,B,C)$

Step 2: Add $f_6(C) = \Sigma_B f_2(B)\, f_5(B,C)$

Remove: $f_2(B)$, $f_5(B,C)$

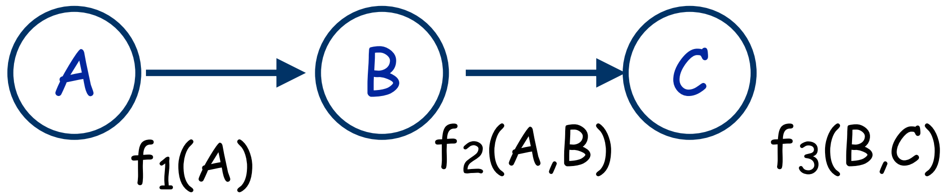
Step 3: Add $f_7(D) = \Sigma_C f_4(C,D)\, f_6(C)$

Remove: $f_4(C,D)$, $f_6(C)$

Last factor $f_7(D)$ is (possibly unnormalized) probability P(D)

Variable Elimination: Evidence

- Computing posterior of query variable given evidence is similar; suppose we observe C=c:



$$
\begin{aligned}
P(A|c) &= \alpha\, P(A)\, P(c|A) \\
&= \alpha\, P(A)\, \Sigma_B\, P(c|B)\, P(B|A) \\
&= \alpha\, f_1(A)\, \Sigma_B\, f_3(B,c)\, f_2(A,B) \\
&= \alpha\, f_1(A)\, \Sigma_B\, f_4(B)\, f_2(A,B) \\
&= \alpha\, f_1(A)\, f_5(A) \\
&= \alpha\, f_6(A)
\end{aligned}
$$

New factors: $f_4(B)= f_3(B,c)$; $f_5(A)= \Sigma_B\, f_2(A,B)\, f_4(B)$; $f_6(A)= f_1(A)\, f_5(A)$

Variable Elimination with Evidence

Given query var Q, evidence vars **E** (observed to be **e**), remaining vars **Z**. Let F be set of factors involving CPTs for {Q} ∪ **Z**.

1. Replace each factor f∈F that mentions a variable(s) in E with its restriction $f_{E=e}$ (somewhat abusing notation)
2. Choose an elimination ordering Z_1, …, Z_n of variables in **Z**.
3. Run variable elimination as above.
4. The remaining factors refer only to the query variable Q. Take their product and normalize to produce P(Q)

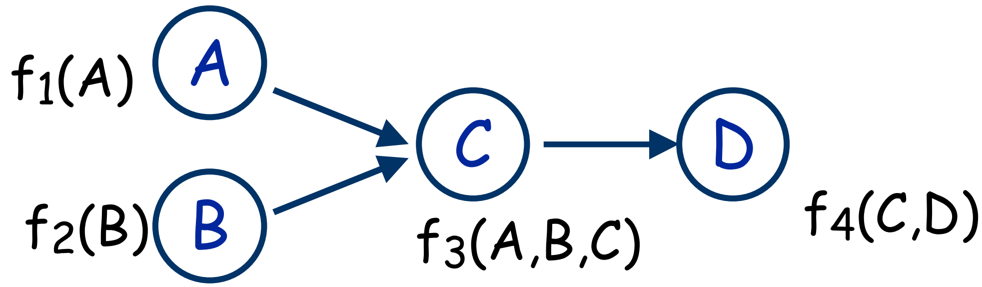
VE: Example 2 again with Evidence

Factors: $f_1(A)$ $f_2(B)$ $f_3(A,B,C)$ $f_4(C,D)$

Query: $P(A)$?

Evidence: $D = d$

Elim. Order: C, B



Restriction: replace $f_4(C,D)$ with $f_5(C) = f_4(C,d)$

Step 1: Add $f_6(A,B) = \Sigma_C\, f_5(C)\, f_3(A,B,C)$

Remove: $f_3(A,B,C)$, $f_5(C)$

Step 2: Add $f_7(A) = \Sigma_B\, f_6(A,B)\, f_2(B)$

Remove: $f_6(A,B)$, $f_2(B)$

Last factors: $f_7(A)$, $f_1(A)$. The product $f_1(A) \times f_7(A)$ is (possibly unnormalized) posterior. So… $P(A|d) = \alpha\, f_1(A) \times f_7(A)$.

Some Notes on the VE Algorithm

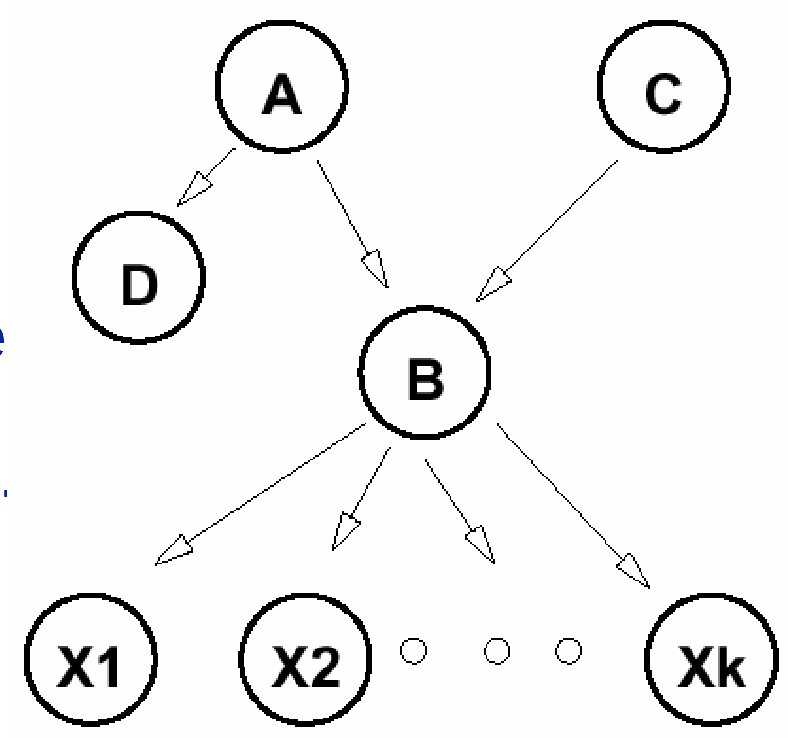
- After iteration j (elimination of Z_j), factors remaining in set F refer only to variables $X_{j+1}, \ldots Z_n$ and Q. No factor mentions an evidence variable E after the initial restriction.
- Number of iterations: linear in number of variables
- Complexity is linear in number of vars and exponential in size of the largest factor. (Recall each factor has exponential size in its number of variables.) Can't do any better than size of BN (since its original factors are part of the factor set). When we create new factors, we might make a set of variables larger.

Some Notes on the VE Algorithm

- The size of the resulting factors is determined by elimination ordering! (We'll see this in detail)
- For *polytrees*, easy to find good ordering (e.g., work outside in).
- For general BNs, sometimes good orderings exist, sometimes they don't (then inference is exponential in number of vars).
  - Simply *finding* the optimal elimination ordering for general BNs is NP-hard.
  - Inference in general is NP-hard in general BNs

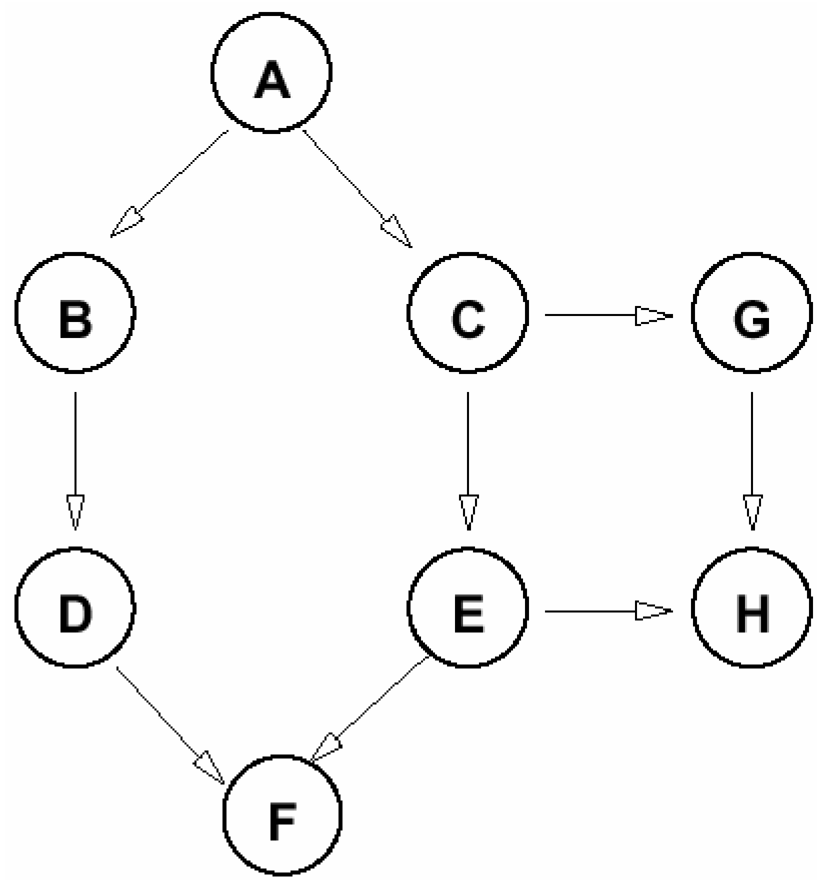
Elimination Ordering: Polytrees

- Inference is linear in size of network
  - ordering: eliminate only "singly-connected" nodes
  - e.g., in this network, eliminate D, A, C, X1,…; or eliminate X1,… Xk, D, A, C; or mix up…
  - result: no factor ever larger than original CPTs
  - eliminating B before these gives factors that include all of A,C, X1,… Xk !!!

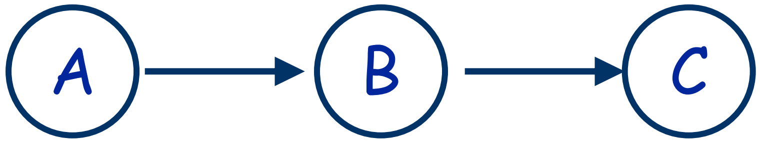


Effect of Different Orderings

- Suppose query variable is D. Consider different orderings for this network
  - A,F,H,G,B,C,E:
    - good: why?
  - E,C,A,B,G,H,F:
    - bad: why?
- Which ordering creates smallest factors?
  - either max size or total
  - which creates largest?



Relevance

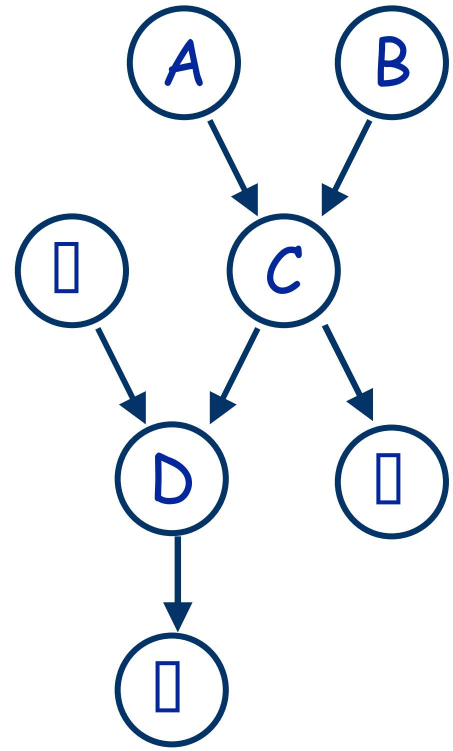


- Certain variables have no impact on the query. In ABC network, computing Pr(A) with no evidence requires elimination of B and C.
  - But when you sum out these vars, you compute a trivial factor (whose value are all ones); for example:
  - eliminating C: $f_4(B) = \Sigma_C f_3(B,C) = \Sigma_C \Pr(C|B)$
  - 1 for any value of B (e.g., Pr(c|b) + Pr(~c|b) = 1)
- No need to think about B or C for this query

Relevance: A Sound Approximation

- Can restrict attention to *relevant* variables. Given query Q, evidence **E**:
  - Q is relevant
  - if any node Z is relevant, its parents are relevant
  - if E∈**E** is a descendent of a relevant node, then E is relevant
- We can restrict our attention to the *subnetwork comprising only relevant variables* when evaluating a query Q

Relevance: Examples

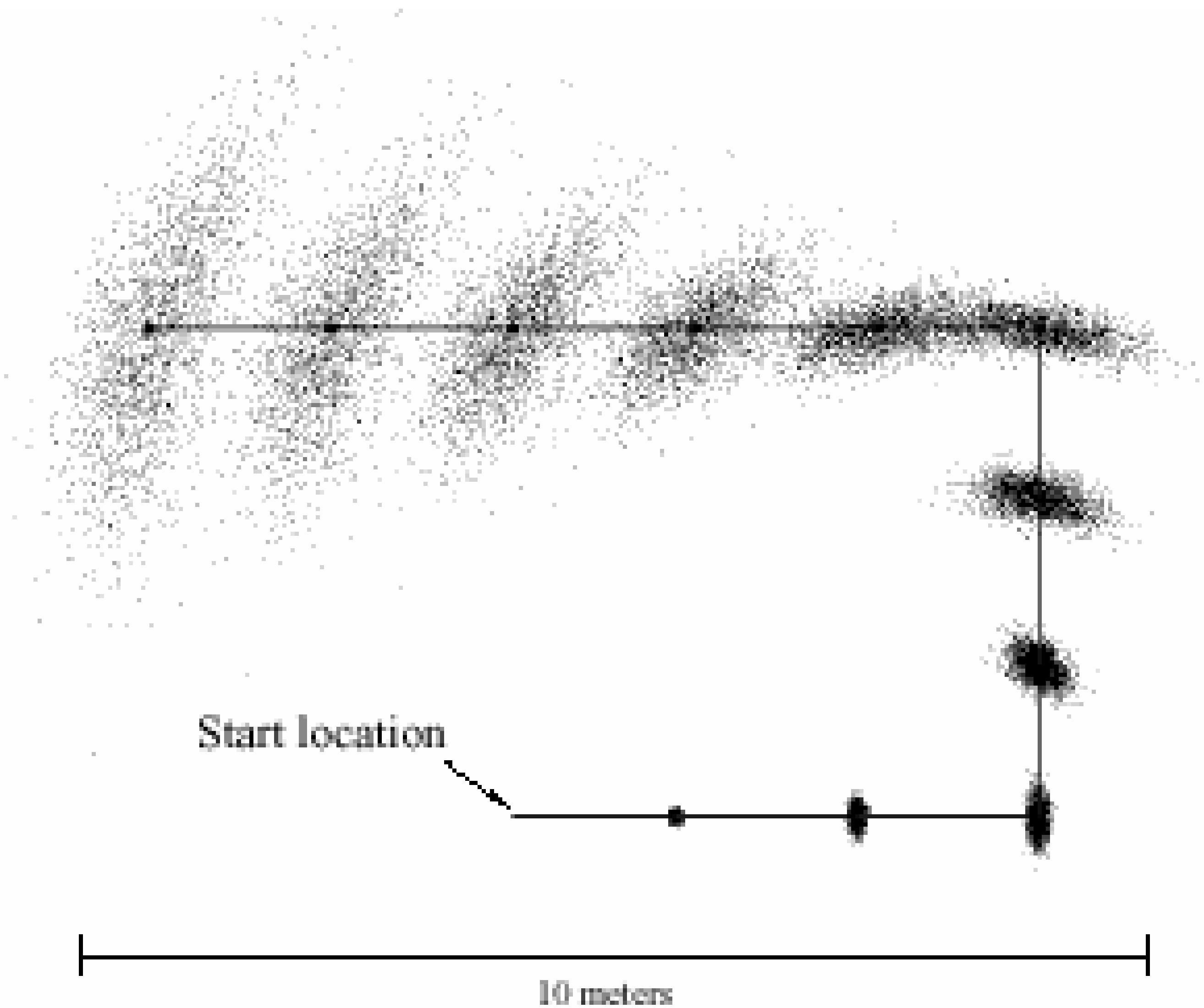


- Query: P(F)
  - relevant: F, C, B, A
- Query: P(F|E)
  - relevant: F, C, B, A
  - also: E, hence D, G
  - intuitively, we need to compute $P(C|E) = \alpha\, P(C)\, P(E|C)$ to accurately compute P(F|E)
- Query: P(F|E,C)
  - algorithm says all vars relevant; but really none except C, F (since C cuts of all influence of others)
  - algorithm is overestimating relevant set

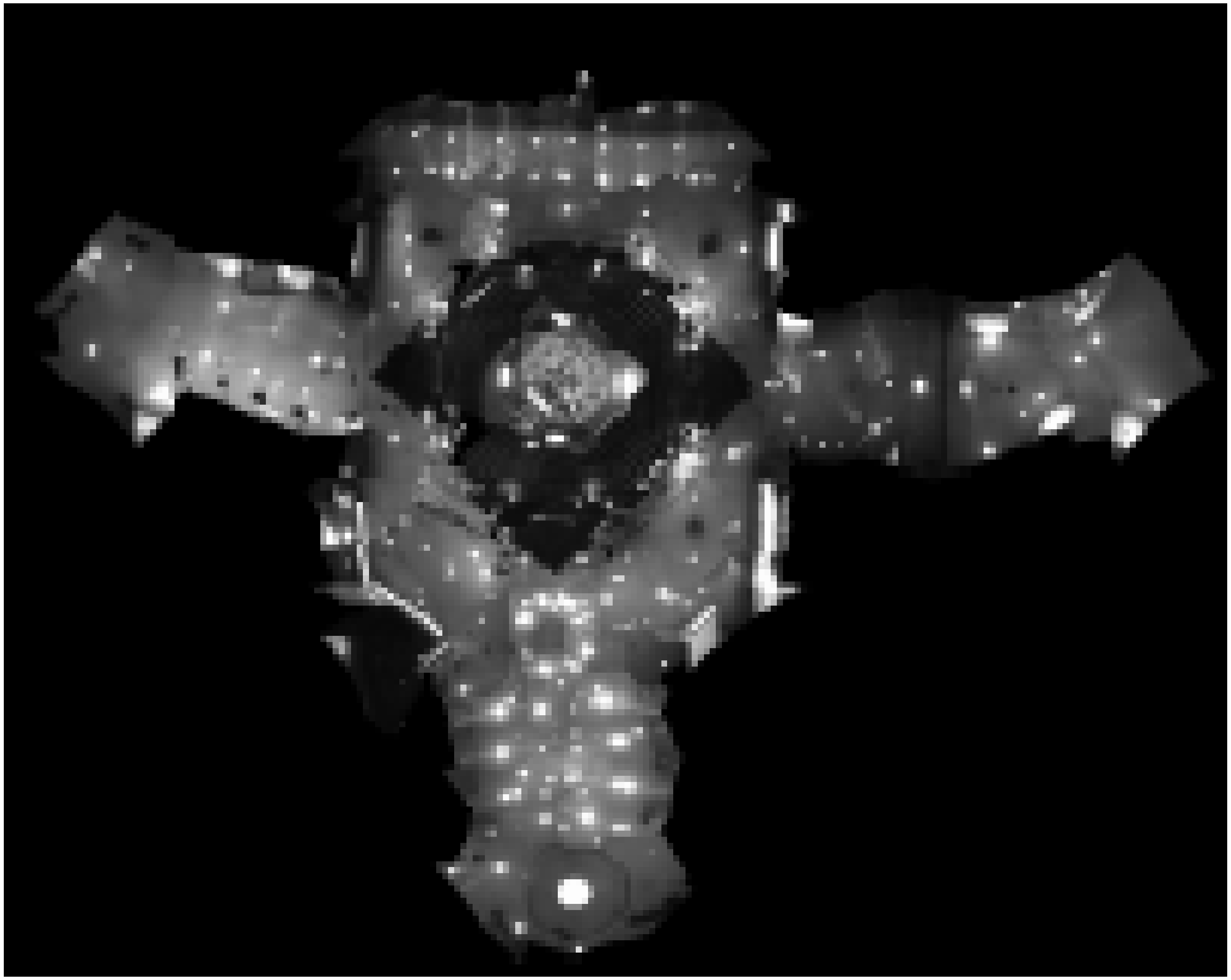
Probabilistic Inference

- Applications of probabilistic inference and Bayesian networks in AI are virtually limitless
- Some examples (slides to follow):
  - mobile robot navigation
  - speech recognition
  - medical diagnosis, patient monitoring
  - process control and monitoring
  - help system under Windows (Bayes nets, really)
  - weather prediction
  - etc.

Robot Uncertainty (Thrun et al.)



Map (Smithsonian: Thrun et al.)

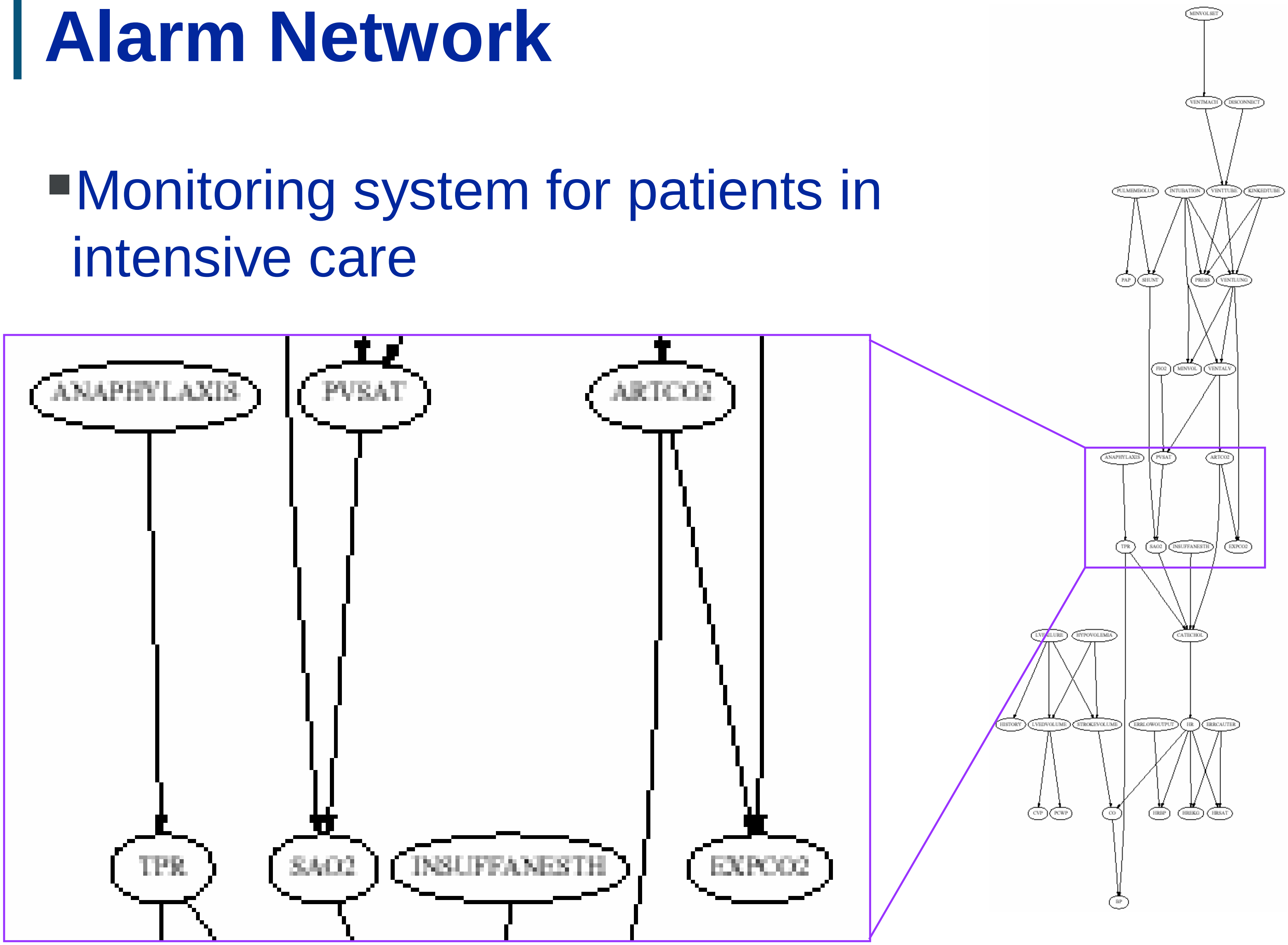


CPSC Network

- BN for diagnosing hepatobiliary diseases
  - 448 nodes, 906 links, 8254 CPT entries
  - see portion of network (not attached)
- Related models (medical diagnosis):
  - QMR
  - Pathfinder
  - etc…

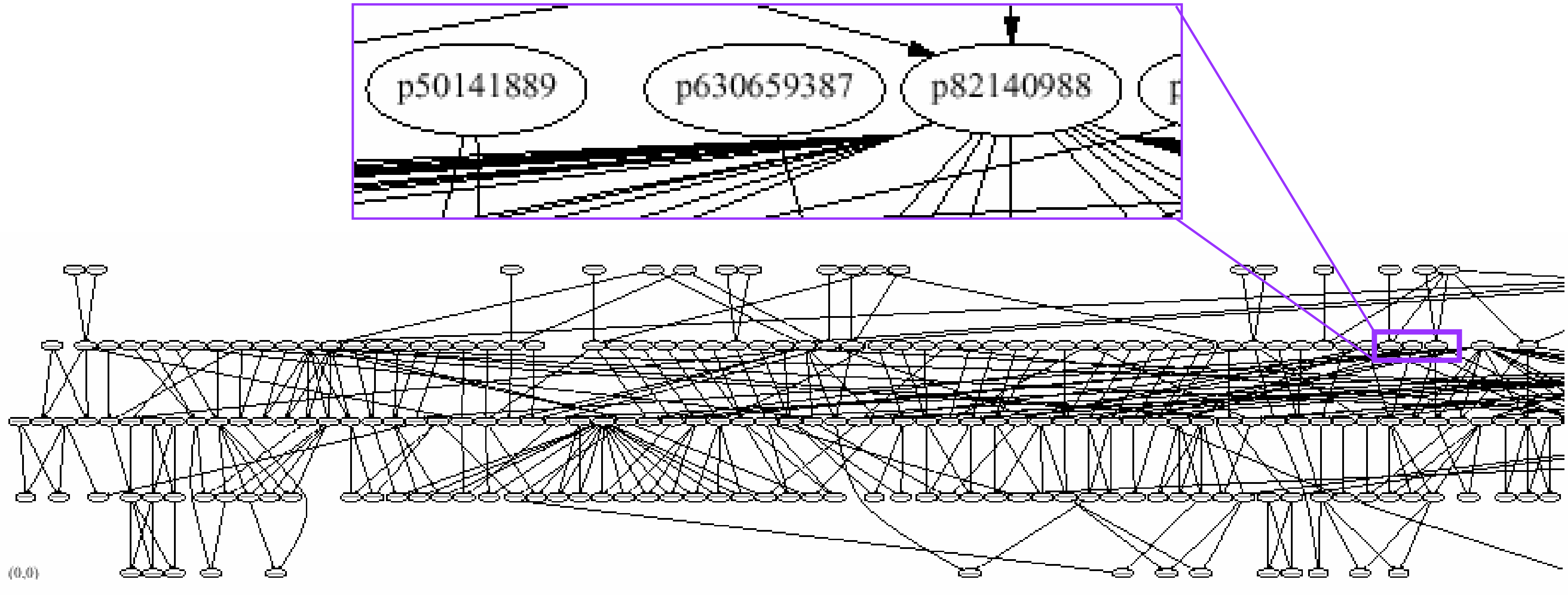
Alarm Network

- Monitoring system for patients in intensive care



Pigs Network

- Determines pedigree of breeding pigs
  - used to diagnose PSE disease
  - half of the network show here



Where Do Bayes Nets Come From?

- Bayes nets often handcrafted
  - interact with a domain expert to: (a) identify dependencies among variables (causal structure); (b) quantify the local distributions (CPTs)
  - empirical data, human expertise often used as guide
- Recent emphasis on learning BNs from data
  - input: a set of *cases* (instantiations of variables)
  - output: a network reflecting empirical distribution
  - issues: identifying causal structure; missing data; discovery of hidden (unobserved) variables; incorporating prior knowledge (bias) about structure

Decision Making under Uncertainty

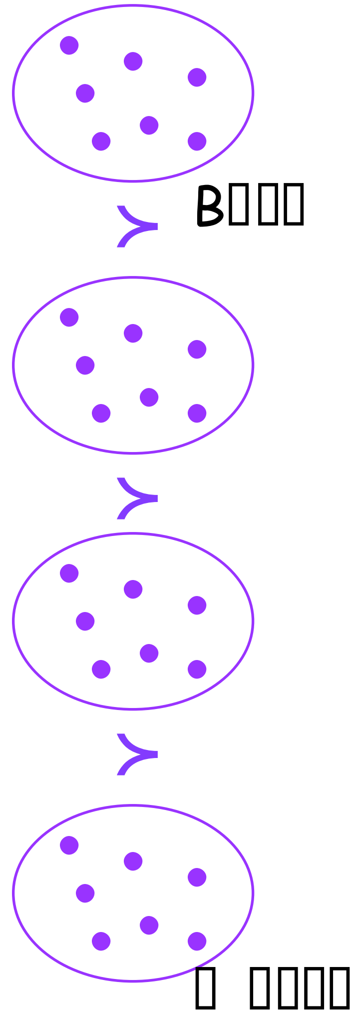
- I give robot a planning problem: I want coffee
  - but coffee maker is broken: robot reports “No plan!”
- If I want more robust behavior – if I want robot to know what to do if my primary goal can’t be satisfied – I should provide it with some indication of my *preferences over alternatives*
  - e.g., coffee better than tea, tea better than water, water better than nothing, etc.
- But it’s more complex:
  - it could wait 45 minutes for coffee maker to be fixed
  - what’s better: tea now? coffee in 45 minutes?
  - could express preferences for <beverage,time> pairs

Preference Orderings

- A *preference ordering* $\succeq$ is a ranking of all possible states of affairs (worlds) S
  - these could be outcomes of actions, truth assts, states in a search problem, etc.
  - $s \succeq t$: means that state s is *at least as good as* t
  - s   t: means that state s is *strictly preferred to* t
- We insist that $\succeq$ is
  - reflexive: i.e., $s \succeq s$ for all states s
  - transitive: i.e., if $s \succeq t$ and $t \succeq w$, then $s \succeq w$
  - connected: for all states s,t, either $s \succeq t$ or $t \succeq s$

Why Impose These Conditions?

- Structure of preference ordering imposes certain "rationality requirements" (it is a weak ordering)
- E.g., why transitivity?
  - Suppose you (strictly) prefer coffee to tea, tea to OJ, OJ to coffee
  - If you prefer X to Y, you'll trade me Y plus $1 for X
  - I can construct a "money pump" and extract arbitrary amounts of money from you

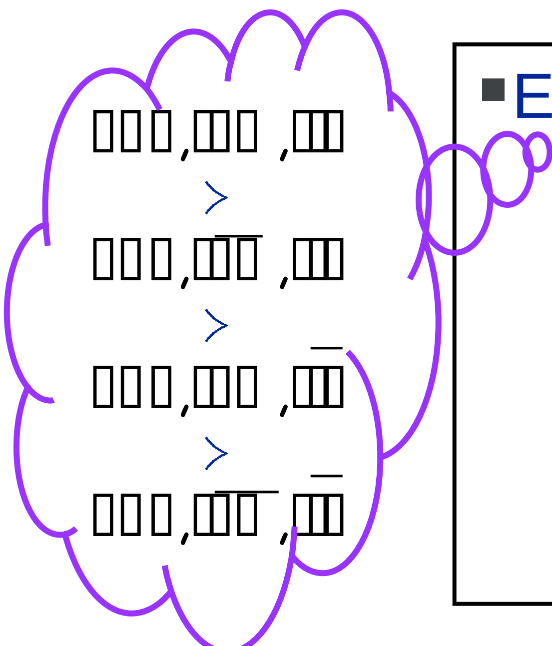


Decision Problems: Certainty

- A *decision problem under certainty* is:
  - a set of *decisions* D
    - e.g., paths in search graph, plans, actions, etc.
  - a set of *outcomes* or states S
    - e.g., states you could reach by executing a plan
  - an *outcome function* $f : D \rightarrow S$
    - the outcome of any decision
  - a preference ordering $\succcurlyeq$ over S
- A *solution* to a decision problem is any $d^* \in D$ such that $f(d^*) \succcurlyeq f(d)$ for all $d \in D$

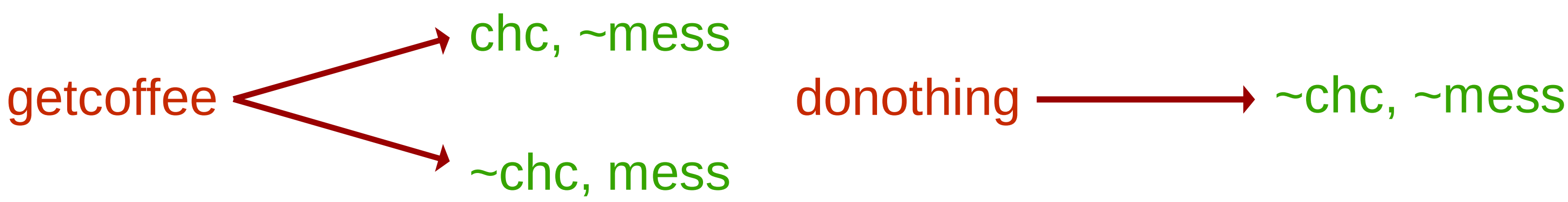
Computational Issues

- At some level, solution to a dec. prob. is trivial
  - complexity lies in the fact that the decisions and outcome function are rarely specified explicitly
  - e.g., in planning or search problem, you *construct* the set of decisions by constructing paths or exploring search paths -- don't know outcomes in advance!



- E.g., our usual office domain
  - We find a plan satisfying chc, ~cm, lt
  - Can we stop searching?
  - Must convince ourselves no better plan exists (nothing can reach best)
  - Generally requires searching entire plan space, unless we have some clever tricks

Decision Making under Uncertainty



- Suppose actions don’t have deterministic outcomes
  - e.g., when robot pours coffee, it spills 20% of time, making a mess
  - preferences: chc, ~mess $\succ$ ~chc,~mess $\succ$ ~chc, mess
- What should robot do?
  - decision *getcoffee* leads to a good outcome and a bad outcome with some probability
  - decision *donothing* leads to a medium outcome for sure
- Should robot be optimistic? pessimistic?
- Really odds of success should influence decision
  - but how?

Utilities

- Rather than just ranking outcomes, we must quantify our degree of preference
  - e.g., how much more important is chc than ~mess
- A *utility function* U:S $\rightarrow \mathbb{R}$ associates a real-valued *utility* with each outcome.
  - U(s) measures your *degree* of preference for s
- Note: U induces a preference ordering $\succcurlyeq_U$ over S defined as: $s \succcurlyeq_U t$ iff $U(s) \geq U(t)$
  - obviously $\succcurlyeq_U$ will be reflexive, transitive, connected

Expected Utility

- Under conditions of uncertainty, each decision d induces a distribution $\Pr_d$ over possible outcomes
  - $\Pr_d(s)$ is probability of outcome s under decision d
- The *expected utility* of decision d is defined

$$EU(d) = \sum_{s \in S} \Pr_d(s) U(s)$$

If U(chc,~ms) = 10, U(~chc,~ms) = 5, U(~chc,ms) = 0,
then EU(getcoffee) = 8 and EU(donothing) = 5

If U(chc,~ms) = 10, U(~chc,~ms) = 9, U(~chc,ms) = 0,
then EU(getcoffee) = 8 and EU(donothing) = 9

The MEU Principle

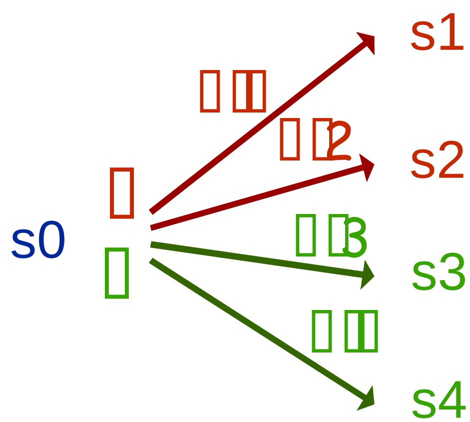
- The *principle of maximum expected utility (MEU)* states that the optimal decision under conditions of uncertainty is that with the greatest expected utility.
- In our example
  - if my utility function is the first one, my robot should get coffee
  - if your utility function is the second one, your robot should do nothing

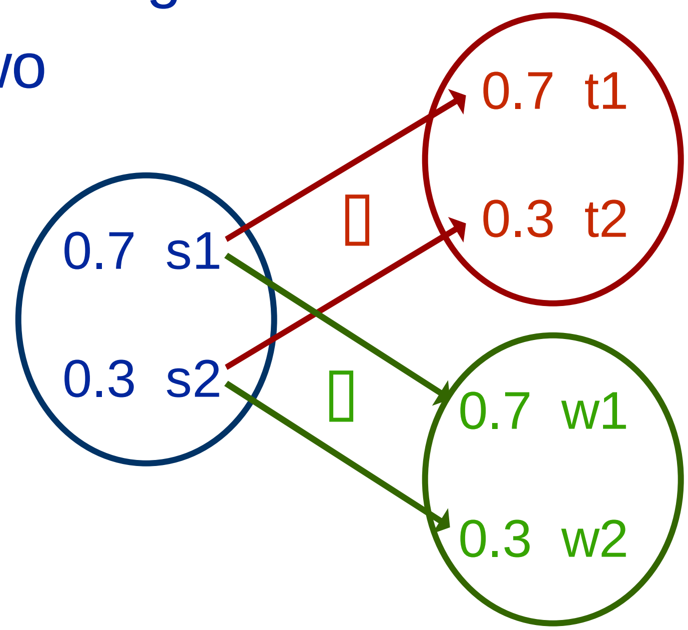
Decision Problems: Uncertainty

- A *decision problem under uncertainty* is:
  - a set of *decisions* D
  - a set of *outcomes* or states S
  - an *outcome function* $Pr : D \rightarrow \Delta(S)$
    - $\Delta(S)$ is the set of distributions over S (e.g., Pr_d)
  - a utility function U over S
- A *solution* to a decision problem under uncertainty is any $d^* \in D$ such that $EU(d^*) \succcurlyeq EU(d)$ for all $d \in D$
- Again, for single-shot problems, this is trivial

Expected Utility: Notes

- Note that this viewpoint accounts for both:
  - uncertainty in action outcomes
  - uncertainty in state of knowledge
  - any combination of the two





Expected Utility: Notes

- Why MEU? Where do utilities come from?
  - underlying foundations of utility theory tightly couple utility with action/choice
  - a utility function can be determined by asking someone about their preferences for actions in specific scenarios (or “lotteries” over outcomes)
- Utility functions needn’t be unique
  - if I multiply U by a positive constant, all decisions have same relative utility
  - if I add a constant to U, same thing
  - *U is unique up to positive affine transformation*

So What are the Complications?

- Outcome space is large
  - like all of our problems, states spaces can be huge
  - don’t want to spell out distributions like Pr_d explicitly
  - Soln: Bayes nets (or related: *influence diagrams*)
- Decision space is large
  - usually our decisions are not one-shot actions
  - rather they involve sequential choices (like plans)
  - if we treat each plan as a distinct decision, decision space is too large to handle directly
  - Soln: use dynamic programming methods to *construct* optimal plans (actually generalizations of plans, called policies… like in game trees)