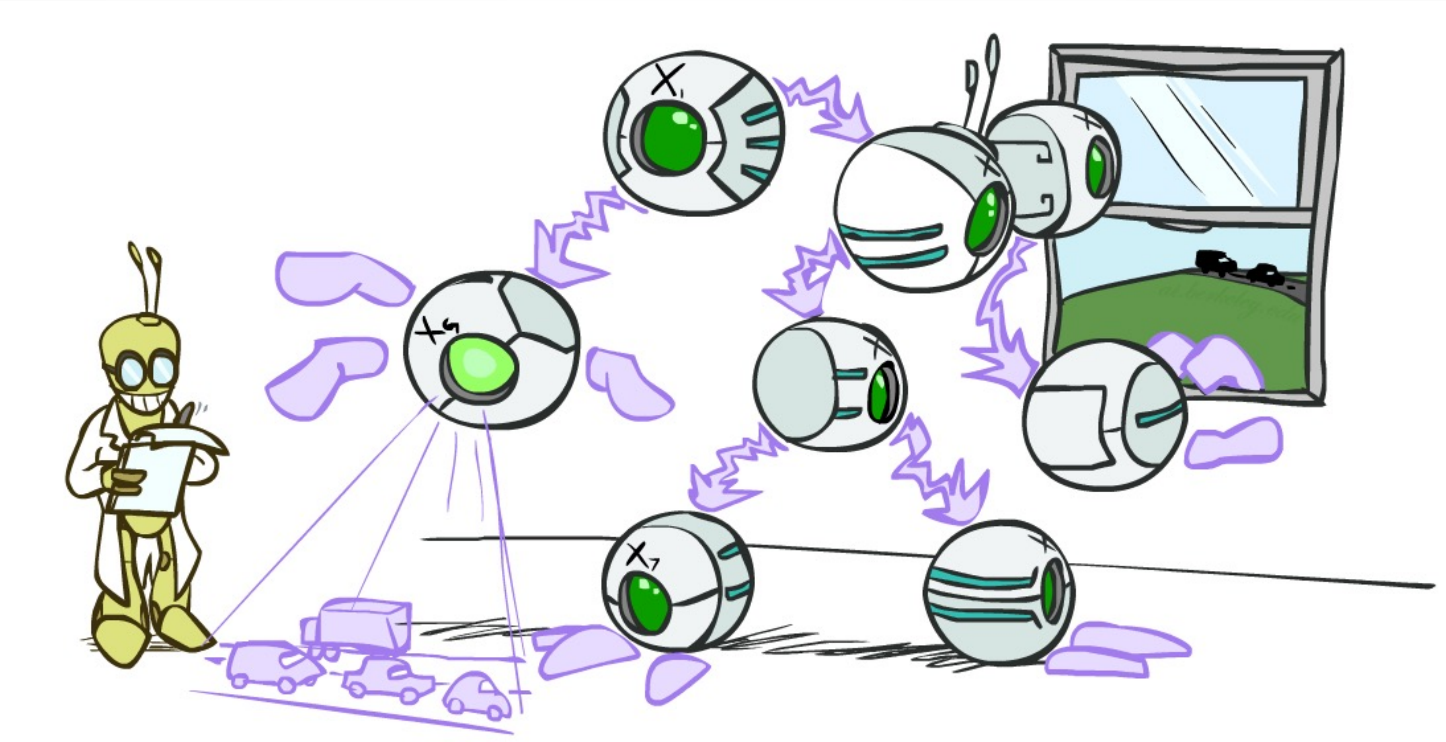
Artificial Intelligence
CE-417, Group 1
Computer Eng. Department
Sharif University of Technology

Fall 2023

By Mohammad Hossein Rohban, Ph.D.

Courtesy: Most slides are adopted from CSE-573 (Washington U.), original slides for the textbook, and CS-188 (UC. Berkeley).

Bayes' Nets: Inference



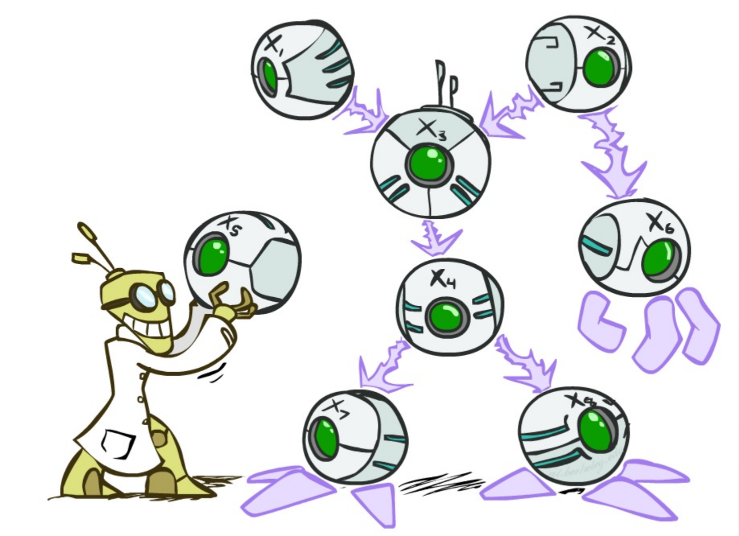
Bayes' Net Representation

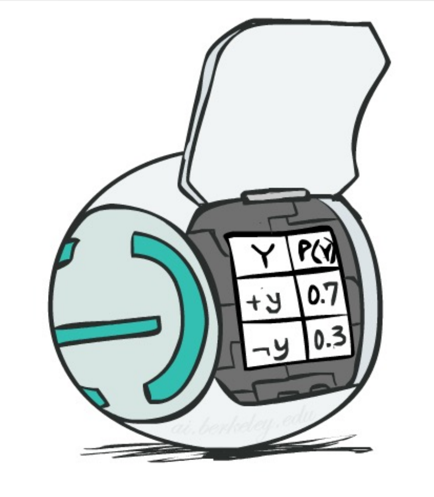
- A directed, acyclic graph, one node per random variable
- A conditional probability table (CPT) for each node
  - A collection of distributions over x, one for each combination of parents' values

$$P(X|a_1 \ldots a_n)$$

- Bayes' nets implicitly encode joint distributions
  - As a product of local conditional distributions
  - To see what probability a BN gives to a full assignment, multiply all the relevant conditionals together:

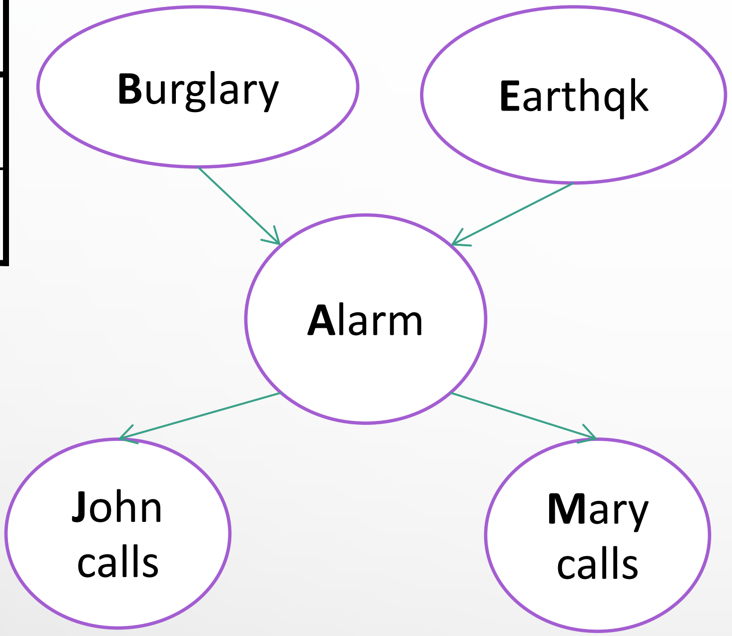
$$P(x_1, x_2, \ldots x_n) = \prod_{i=1}^{n} P(x_i|parents(X_i))$$



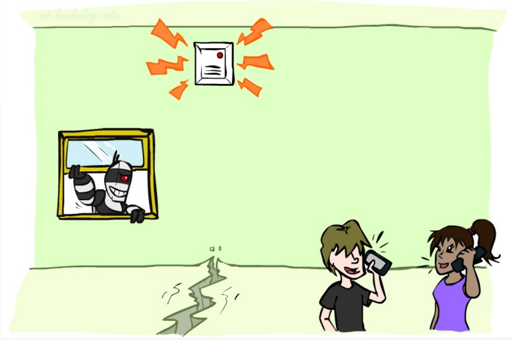


Example: Alarm Network

B	P(B)
+b	0.001
-b	0.999



E	P(E)
+e	0.002
-e	0.998



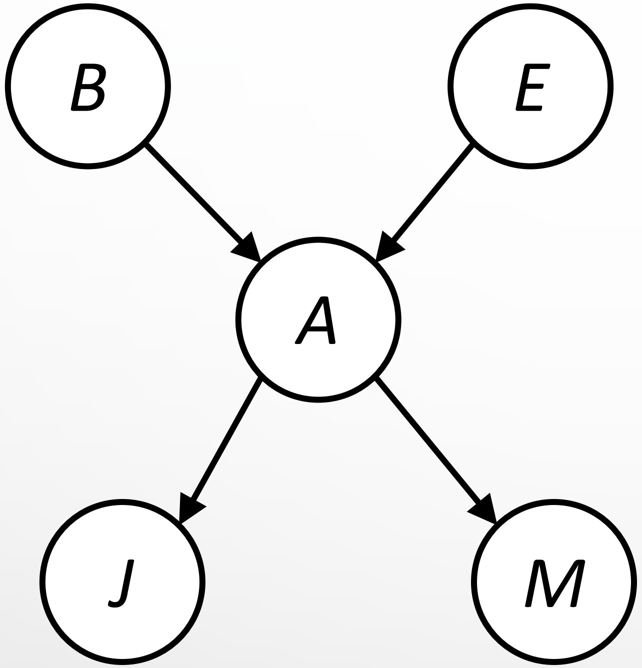
A	J	P(J|A)
+a	+j	0.9
+a	-j	0.1
-a	+j	0.05
-a	-j	0.95

A	M	P(M|A)
+a	+m	0.7
+a	-m	0.3
-a	+m	0.01
-a	-m	0.99

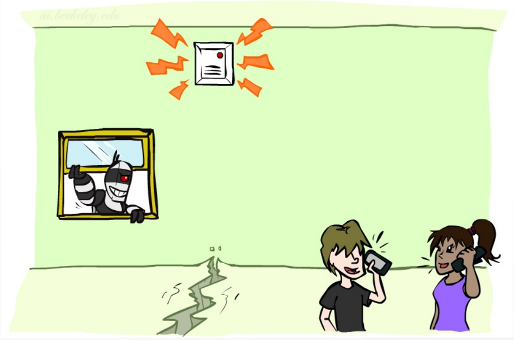
B	E	A	P(A|B,E)
+b	+e	+a	0.95
+b	+e	-a	0.05
+b	-e	+a	0.94
+b	-e	-a	0.06
-b	+e	+a	0.29
-b	+e	-a	0.71
-b	-e	+a	0.001
-b	-e	-a	0.999

Example: Alarm Network

B	P(B)
+b	0.001
-b	0.999



E	P(E)
+e	0.002
-e	0.998



A	J	P(J|A)
+a	+j	0.9
+a	-j	0.1
-a	+j	0.05
-a	-j	0.95

A	M	P(M|A)
+a	+m	0.7
+a	-m	0.3
-a	+m	0.01
-a	-m	0.99

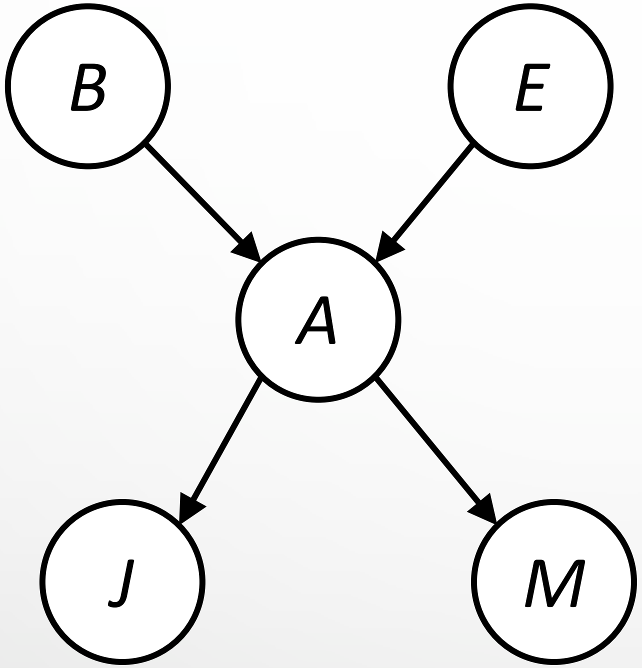
B	E	A	P(A|B,E)
+b	+e	+a	0.95
+b	+e	-a	0.05
+b	-e	+a	0.94
+b	-e	-a	0.06
-b	+e	+a	0.29
-b	+e	-a	0.71
-b	-e	+a	0.001
-b	-e	-a	0.999

$$P(+b, -e, +a, -j, +m) =$$

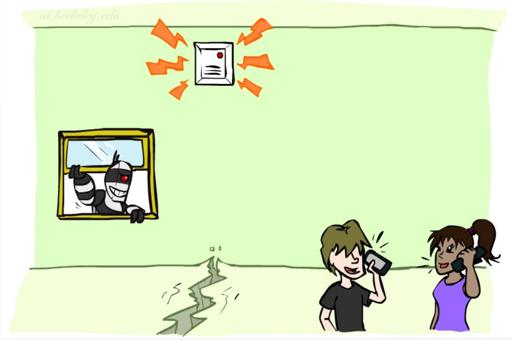
$$P(+b)P(-e)P(+a| + b, -e)P(-j| + a)P(+m| + a) =$$

Example: Alarm Network

B	P(B)
+b	0.001
-b	0.999



E	P(E)
+e	0.002
-e	0.998



A	J	P(J|A)
+a	+j	0.9
+a	-j	0.1
-a	+j	0.05
-a	-j	0.95

A	M	P(M|A)
+a	+m	0.7
+a	-m	0.3
-a	+m	0.01
-a	-m	0.99

B	E	A	P(A|B,E)
+b	+e	+a	0.95
+b	+e	-a	0.05
+b	-e	+a	0.94
+b	-e	-a	0.06
-b	+e	+a	0.29
-b	+e	-a	0.71
-b	-e	+a	0.001
-b	-e	-a	0.999

$$P(+b, -e, +a, -j, +m) =$$

$$P(+b)P(-e)P(+a| + b, -e)P(-j| + a)P(+m| + a) =$$

$$0.001 \times 0.998 \times 0.94 \times 0.1 \times 0.7$$

Bayes' Nets

- ✔ Representation
- ✔ Conditional independences
- Probabilistic inference
  - Enumeration (exact, exponential complexity)
  - Variable elimination (exact, worst-case exponential complexity, often better)
  - Inference is NP-complete
  - Sampling (approximate)
- Learning Bayes' Nets from data

Inference

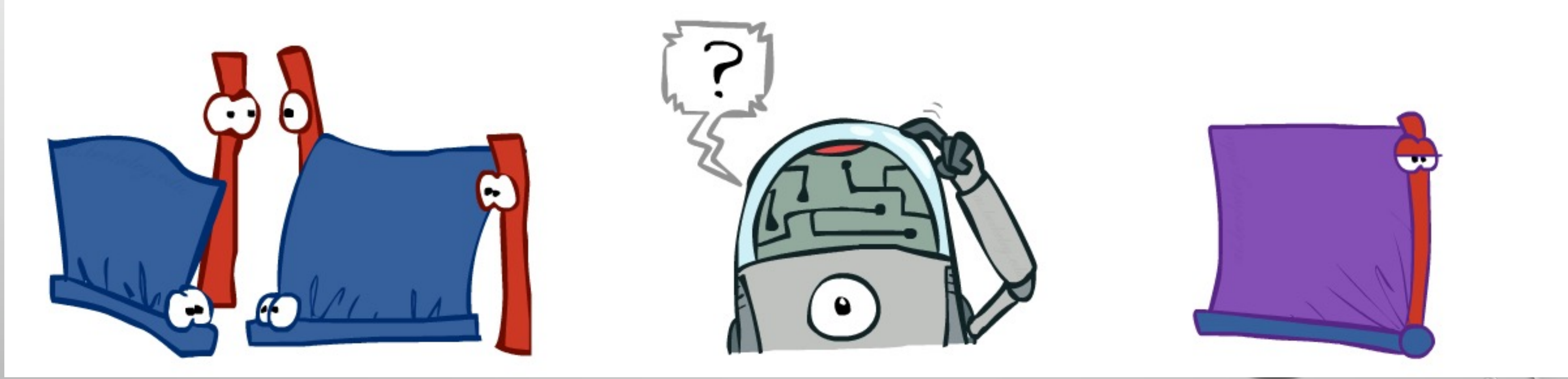
- Inference: calculating some useful quantity from a joint probability distribution

- Examples:
  - Posterior probability

    $P(Q|E_1 = e_1, \ldots E_k = e_k)$

  - Most likely explanation:

    $\text{argmax}_q \; P(Q = q|E_1 = e_1 \ldots)$



Inference by Enumeration

- General case:
  - Evidence variables: $E_1 \dots E_k = e_1 \dots e_k$
  - Query* variable: Q
  - Hidden variables: $H_1 \dots H_r$

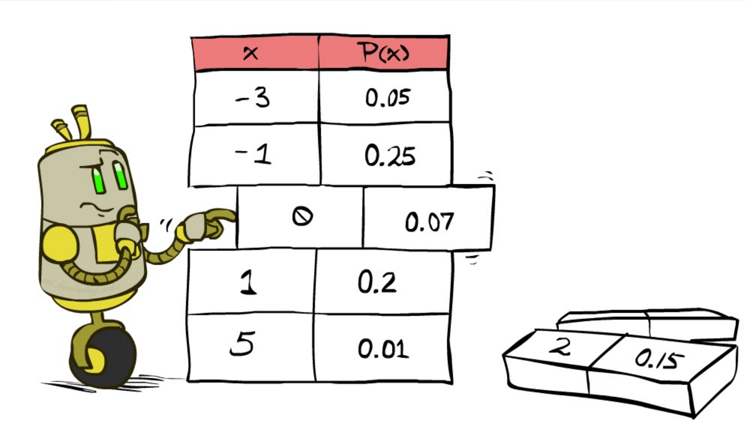
$\Big\}$ $X_1, X_2, \dots X_n$ *All variables*

- We want:

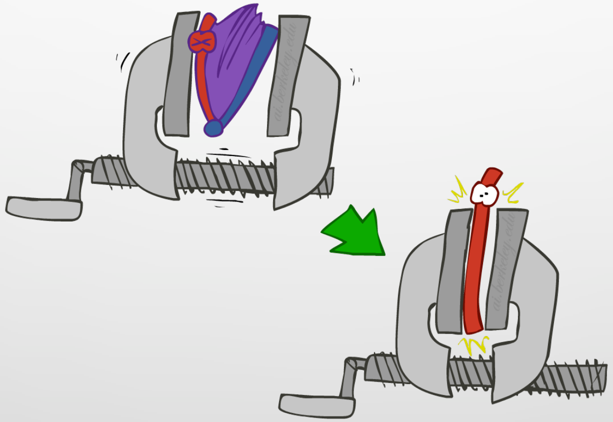
** Works fine with multiple query variables, too*

$$P(Q|e_1 \dots e_k)$$

- Step 1: Select the entries consistent with the evidence



- Step 2: Sum out H to get joint of Query and evidence



$$P(Q, e_1 \dots e_k) = \sum_{h_1 \dots h_r} \underbrace{P(Q, h_1 \dots h_r, e_1 \dots e_k)}_{X_1, X_2, \dots X_n}$$

- Step 3: Normalize

$$\times \frac{1}{Z}$$

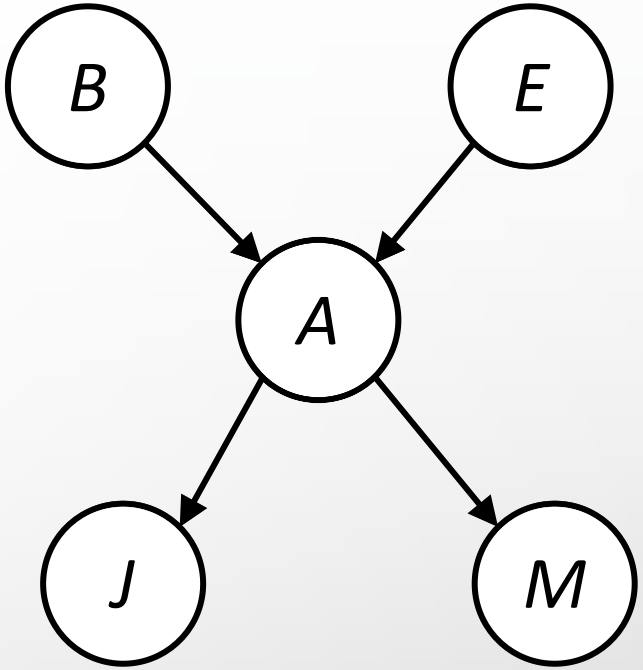
$$Z = \sum_q P(Q, e_1 \cdots e_k)$$

$$P(Q|e_1 \cdots e_k) = \frac{1}{Z} P(Q, e_1 \cdots e_k)$$

Inference by Enumeration in Bayes' Net

- Given unlimited time, inference in BNs is easy

- Reminder of inference by enumeration by example:



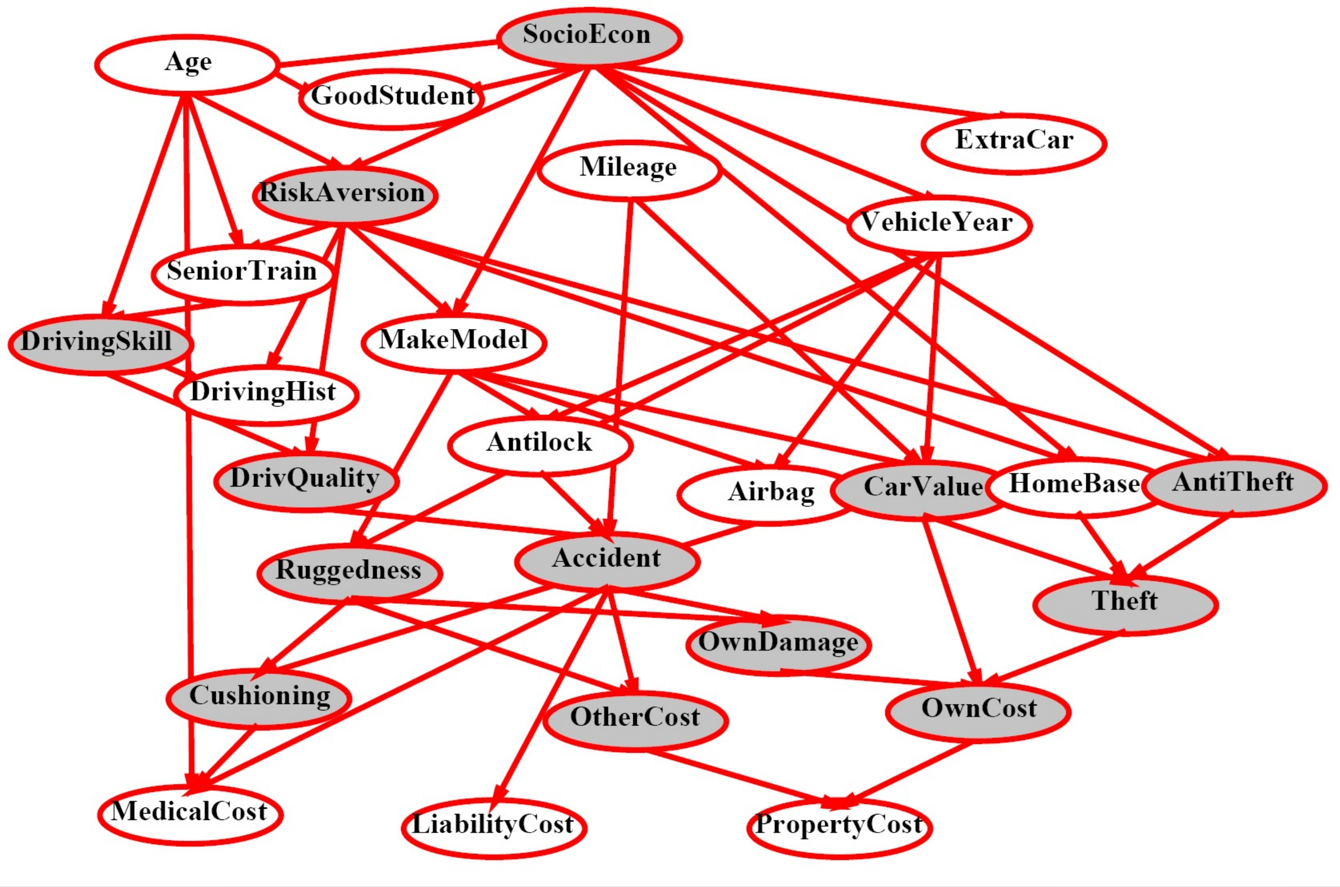
$$P(B \mid +j, +m) \propto_B P(B, +j, +m)$$

$$= \sum_{e,a} P(B, e, a, +j, +m)$$

$$= \sum_{e,a} P(B)P(e)P(a|B,e)P(+j|a)P(+m|a)$$

$$= P(B)P(+e)P(+a|B,+e)P(+j|+a)P(+m|+a) + P(B)P(+e)P(-a|B,+e)P(+j|-a)P(+m|-a)$$
$$P(B)P(-e)P(+a|B,-e)P(+j|+a)P(+m|+a) + P(B)P(-e)P(-a|B,-e)P(+j|-a)P(+m|-a)$$

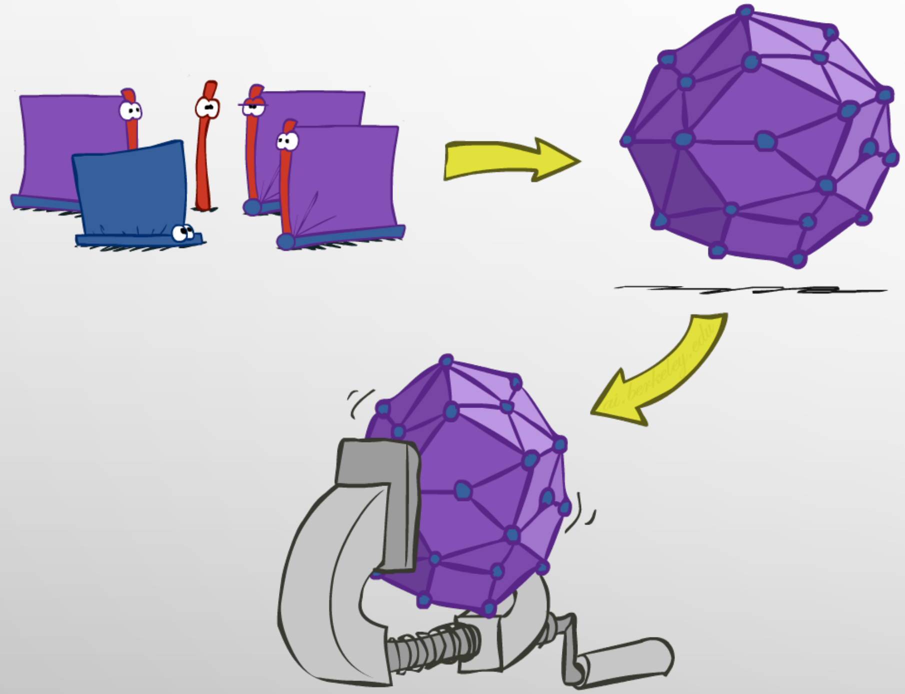
Inference by Enumeration?



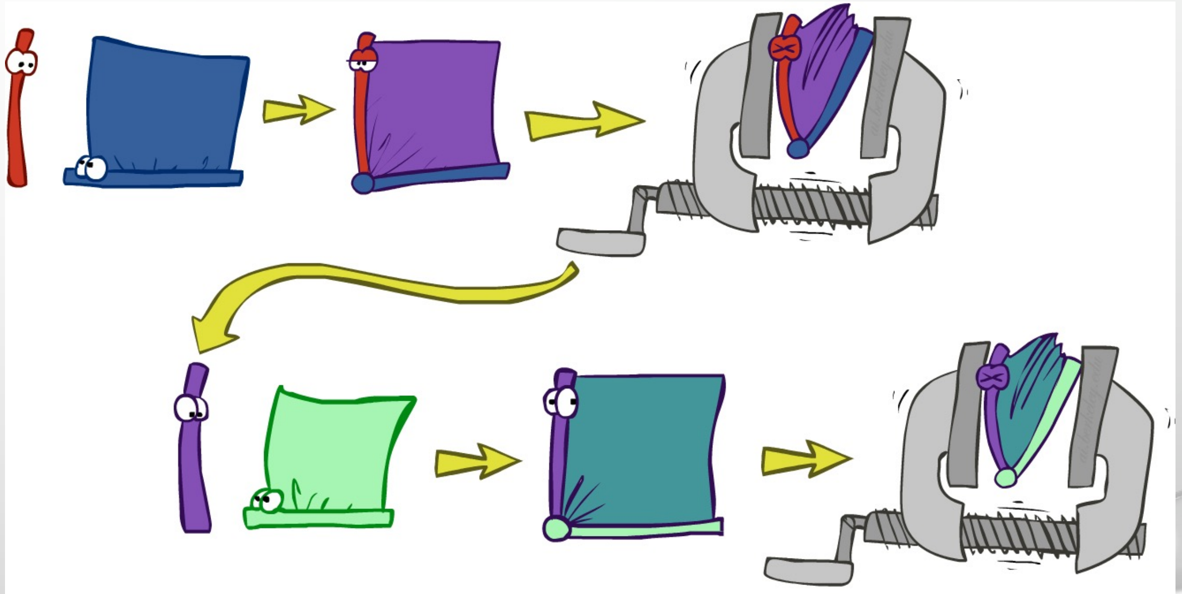
$$P(Antilock|observed\ variables) = ?$$

Inference by Enumeration vs. Variable Elimination

- Why is inference by enumeration so slow?
  - You join up the whole joint distribution before you sum out the hidden variables



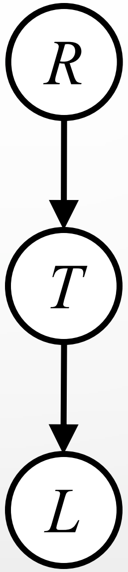
- Idea: interleave joining and marginalizing!
  - Called "Variable Elimination"
  - Still NP-hard, but usually much faster than inference by enumeration



  - First we'll need some new notation: factors

Example: Traffic Domain

- Random variables
  - R: Raining
  - T: Traffic
  - L: Late for class!



$P(R)$

+r	0.1
-r	0.9

$P(T|R)$

+r	+t	0.8
+r	-t	0.2
-r	+t	0.1
-r	-t	0.9

$P(L|T)$

+t	+l	0.3
+t	-l	0.7
-t	+l	0.1
-t	-l	0.9

$$P(L) = ?$$

$$= \sum_{r,t} P(r, t, L)$$

$$= \sum_{r,t} P(r)P(t|r)P(L|t)$$

Inference by Enumeration: Procedural Outline

- Track objects called factors
- Initial factors are local CPTs (one per node)

$P(R)$

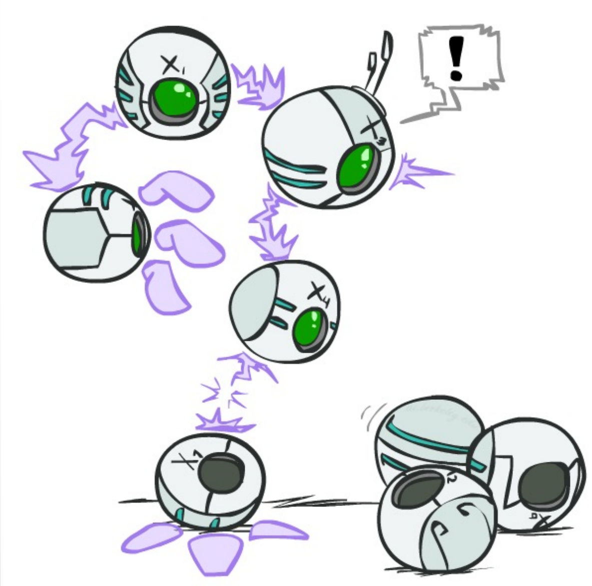
+r	0.1
-r	0.9

$P(T|R)$

+r	+t	0.8
+r	-t	0.2
-r	+t	0.1
-r	-t	0.9

$P(L|T)$

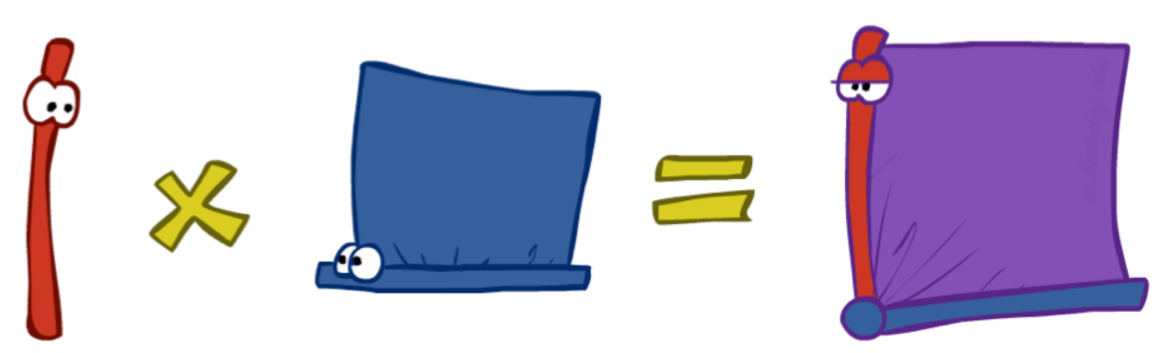
+t	+l	0.3
+t	-l	0.7
-t	+l	0.1
-t	-l	0.9



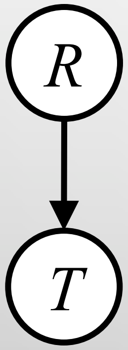
Procedure: join all factors, then eliminate all hidden variables

Operation 1: Join Factors

- First basic operation: joining factors
- Combining factors:
  - Just like a database join
  - Get all factors over the joining variable
  - Build a new factor over the union of the variables involved



- Example: join on R

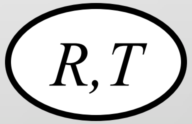


$P(R) \times P(T|R) \rightarrow P(R,T)$

+r	0.1
-r	0.9

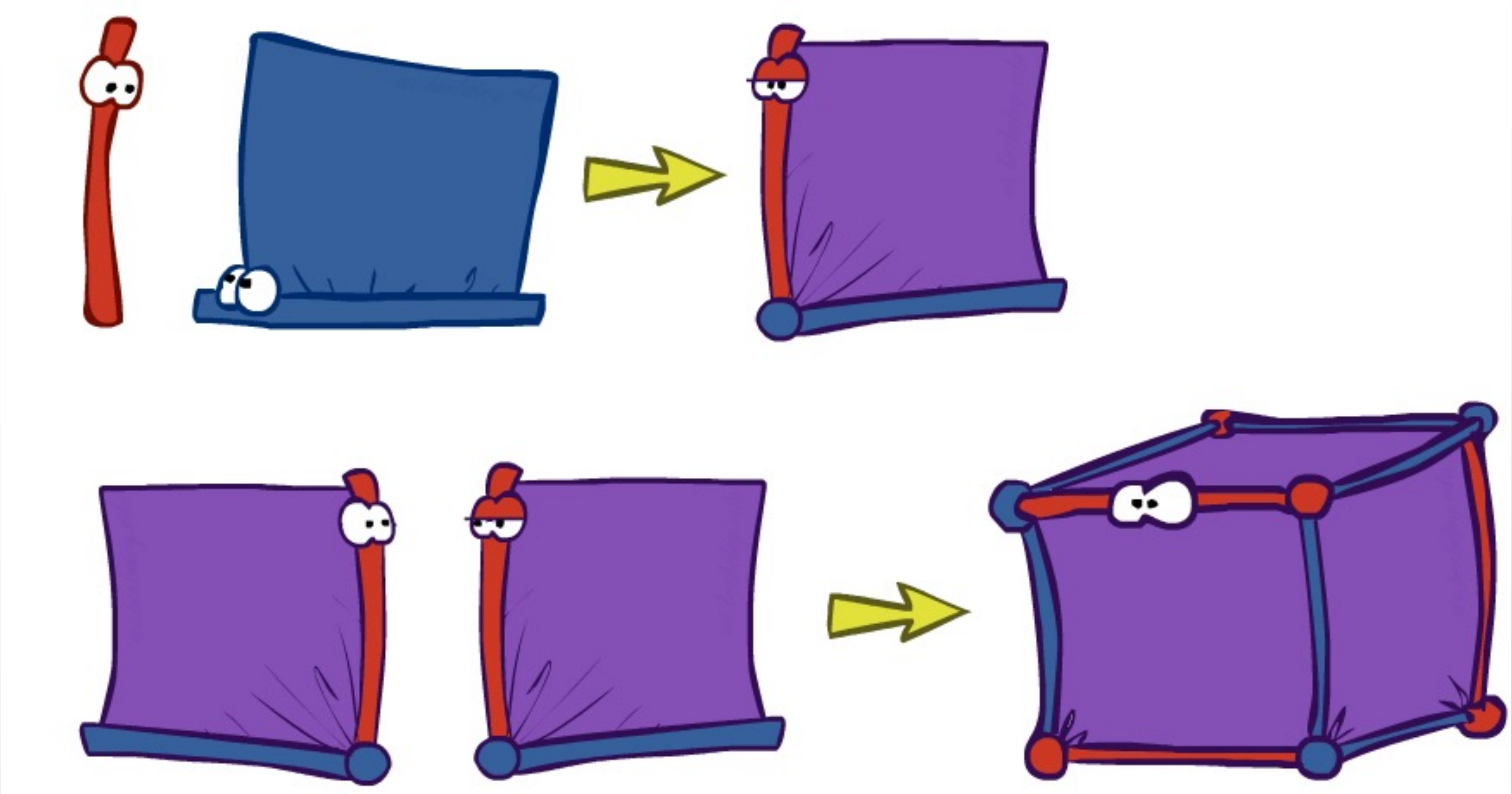
+r	+t	0.8
+r	-t	0.2
-r	+t	0.1
-r	-t	0.9

+r	+t	0.08
+r	-t	0.02
-r	+t	0.09
-r	-t	0.81

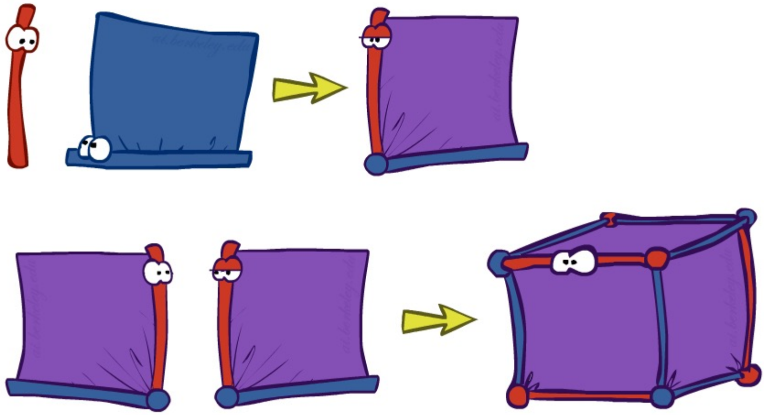


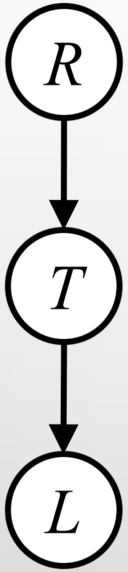
  - Computation for each entry: pointwise products $\forall r, t : \quad P(r,t) = P(r) \cdot P(t|r)$

Example: Multiple Joins



Example: Multiple Joins





$P(R)$

+r	0.1
-r	0.9

$P(T|R)$

+r	+t	0.8
+r	-t	0.2
-r	+t	0.1
-r	-t	0.9

$P(L|T)$

+t	+l	0.3
+t	-l	0.7
-t	+l	0.1
-t	-l	0.9

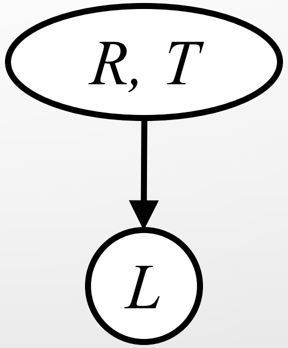
Join R

$P(R,T)$

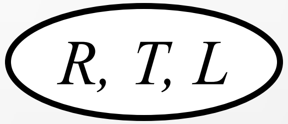
+r	+t	0.08
+r	-t	0.02
-r	+t	0.09
-r	-t	0.81

$P(L|T)$

+t	+l	0.3
+t	-l	0.7
-t	+l	0.1
-t	-l	0.9



Join T



$P(R,T,L)$

+r	+t	+l	0.024
+r	+t	-l	0.056
+r	-t	+l	0.002
+r	-t	-l	0.018
-r	+t	+l	0.027
-r	+t	-l	0.063
-r	-t	+l	0.081
-r	-t	-l	0.729

Operation 2: Eliminate

- Second basic operation: marginalization
- Take a factor and sum out a variable
  - Shrinks a factor to a smaller one
  - A projection operation
- Example:

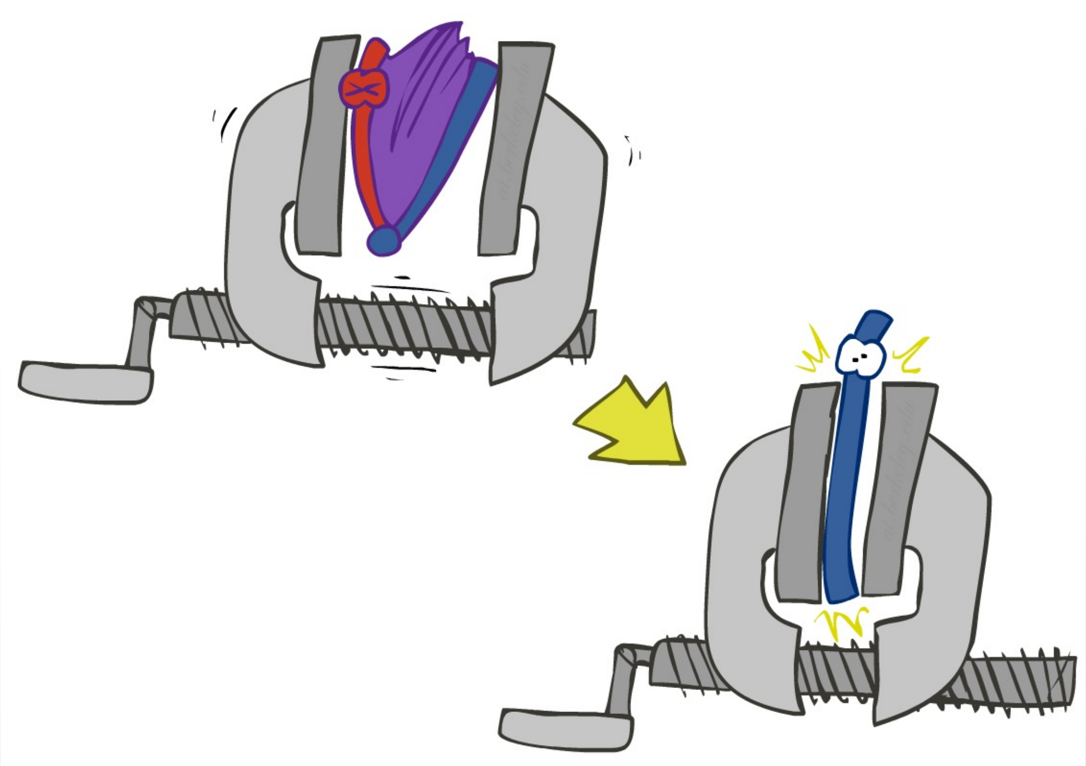
$P(R,T)$

+r	+t	0.08
+r	-t	0.02
-r	+t	0.09
-r	-t	0.81

sum R →

$P(T)$

+t	0.17
-t	0.83



Multiple Elimination

R, T, L → Sum out R → *T, L* → Sum out T → *L*

$P(R,T,L)$

+r	+t	+l	0.024
+r	+t	-l	0.056
+r	-t	+l	0.002
+r	-t	-l	0.018
-r	+t	+l	0.027
-r	+t	-l	0.063
-r	-t	+l	0.081
-r	-t	-l	0.729

Sum out R

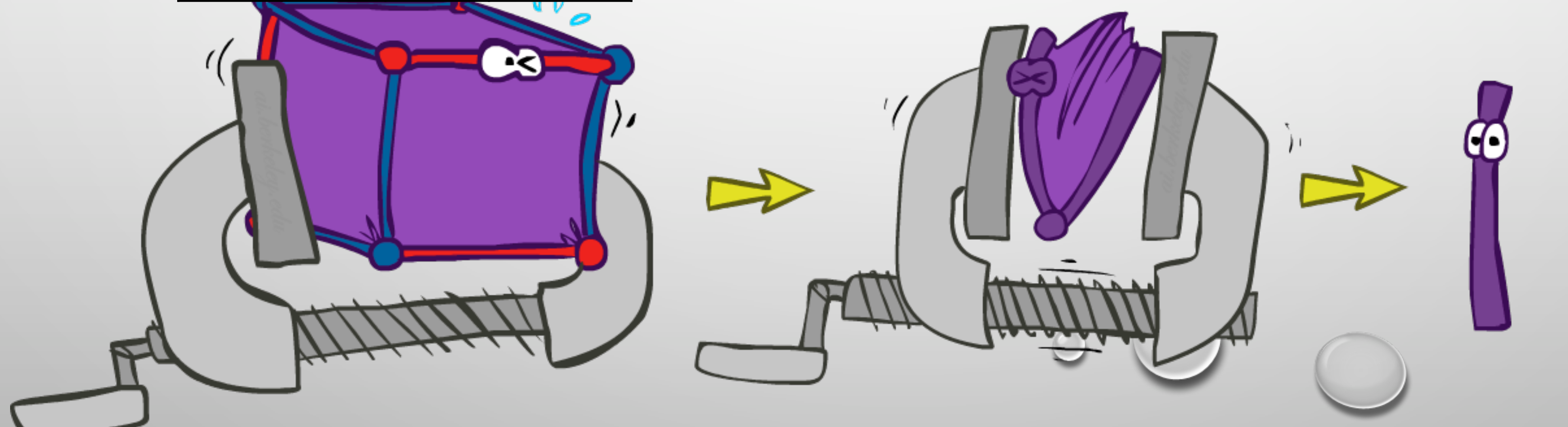
$P(T,L)$

+t	+l	0.051
+t	-l	0.119
-t	+l	0.083
-t	-l	0.747

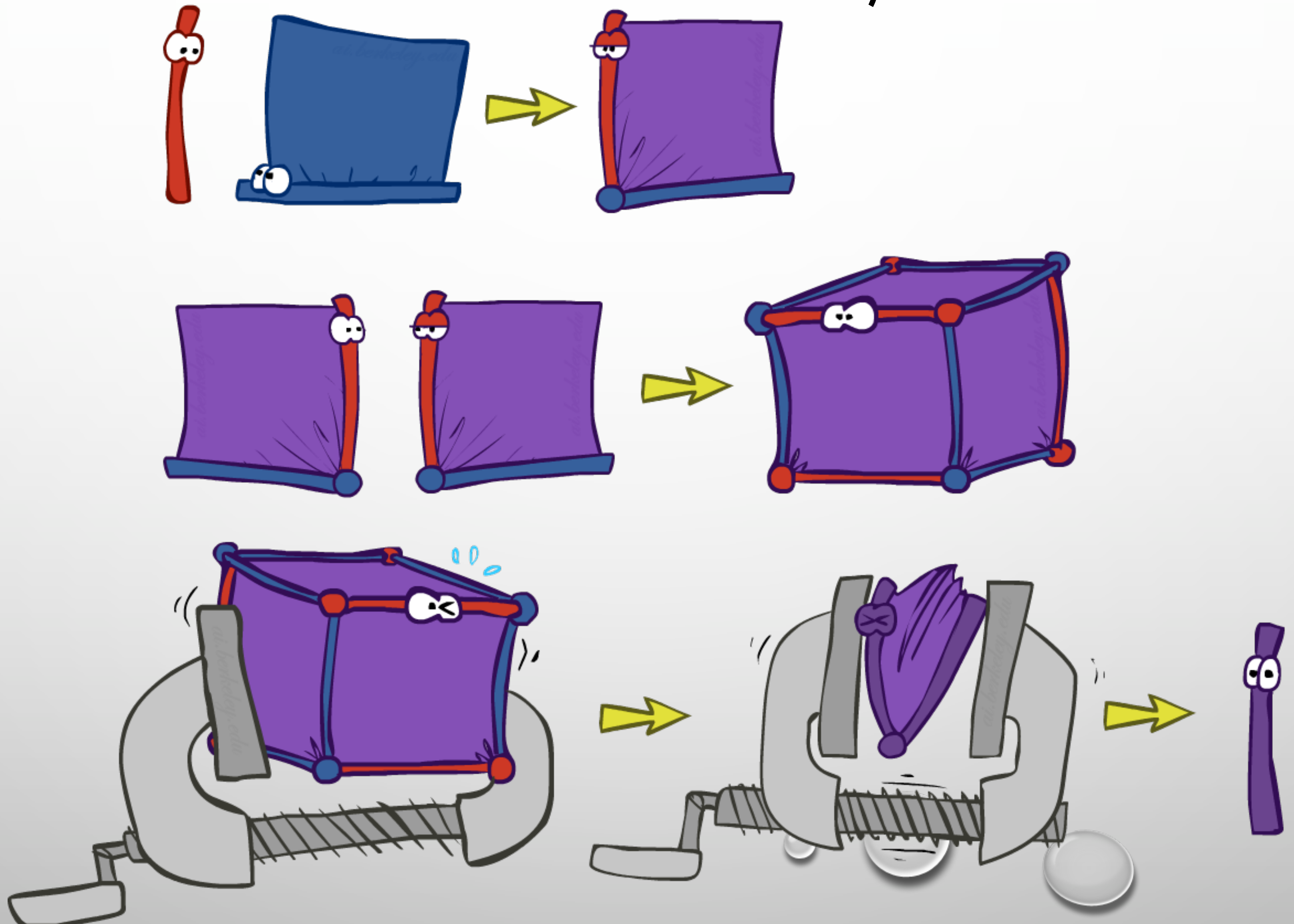
Sum out T

$P(L)$

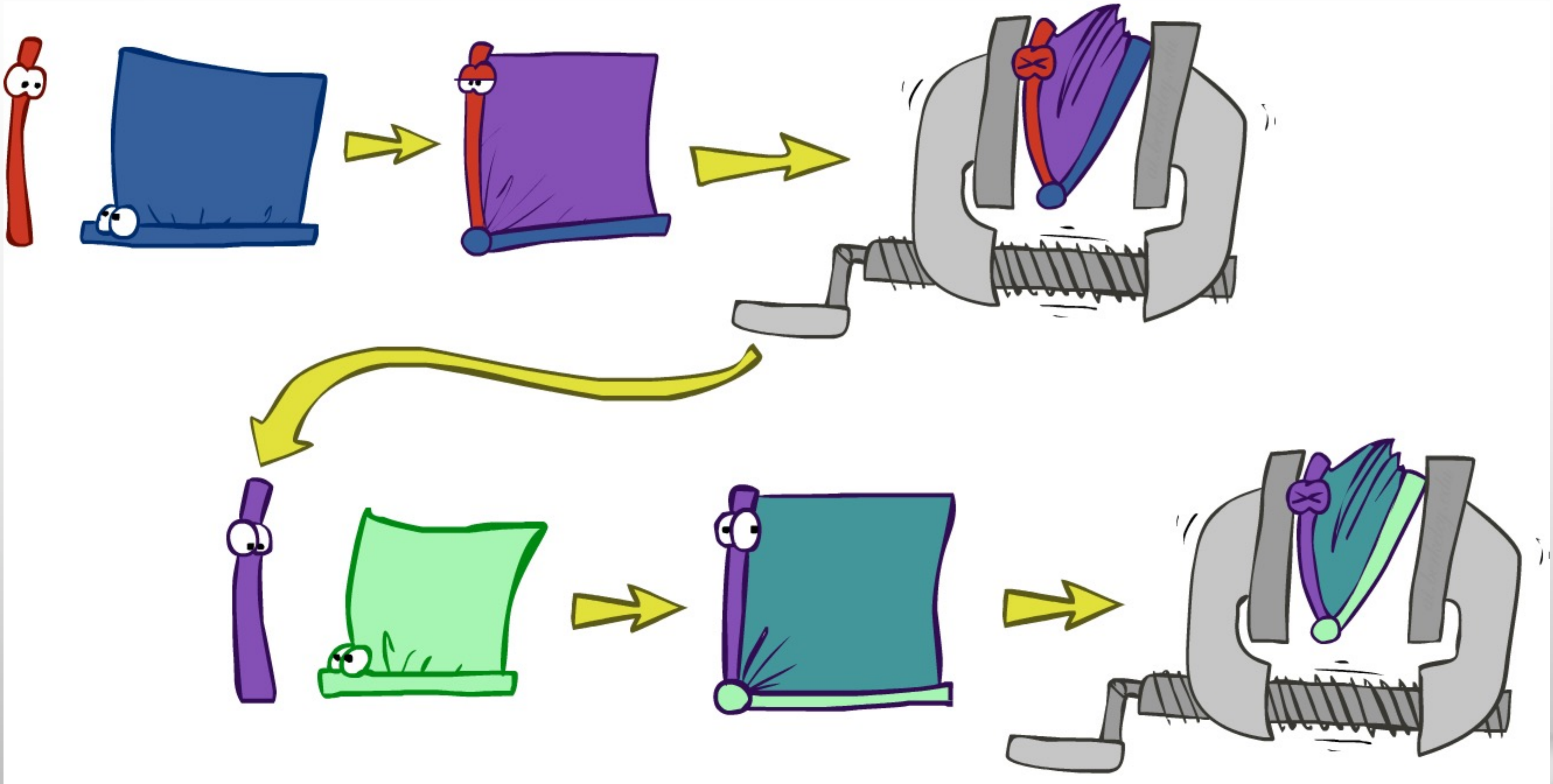
+l	0.134
-l	0.886



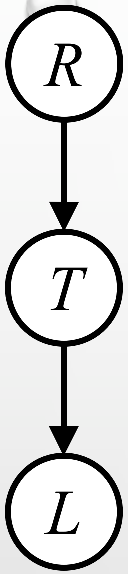
Thus Far: Multiple Join, Multiple Eliminate (= Inference by Enumeration)



Marginalizing Early (= Variable Elimination)



Traffic Domain



$$P(L) = ?$$

- Inference by enumeration

$$= \sum_t \sum_r P(L|t)P(r)P(t|r)$$

Join on r

Join on t

Eliminate r

Eliminate t

- Variable Elimination

$$= \sum_t P(L|t) \sum_r P(r)P(t|r)$$

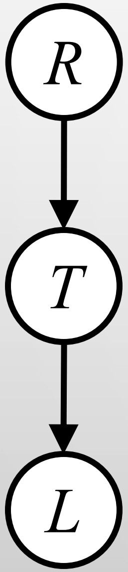
Join on r

Eliminate r

Join on t

Eliminate t

Marginalizing Early! (aka VE)



$P(R)$

+r	0.1
-r	0.9

$P(T|R)$

+r	+t	0.8
+r	-t	0.2
-r	+t	0.1
-r	-t	0.9

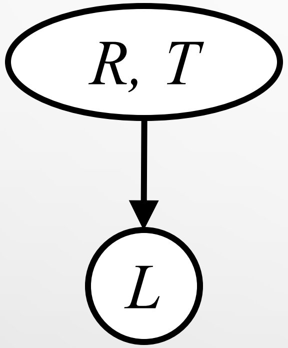
$P(L|T)$

+t	+l	0.3
+t	-l	0.7
-t	+l	0.1
-t	-l	0.9

Join R

$P(R,T)$

+r	+t	0.08
+r	-t	0.02
-r	+t	0.09
-r	-t	0.81



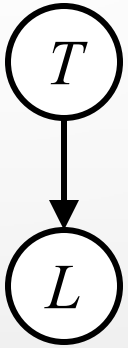
$P(L|T)$

+t	+l	0.3
+t	-l	0.7
-t	+l	0.1
-t	-l	0.9

Sum out R

$P(T)$

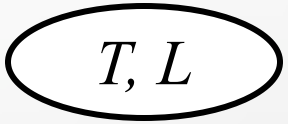
+t	0.17
-t	0.83



$P(L|T)$

+t	+l	0.3
+t	-l	0.7
-t	+l	0.1
-t	-l	0.9

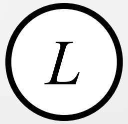
Join T



$P(T,L)$

+t	+l	0.051
+t	-l	0.119
-t	+l	0.083
-t	-l	0.747

Sum out T



$P(L)$

+l	0.134
-l	0.866

Evidence

- If evidence, start with factors that select that evidence
  - No evidence uses these initial factors:

$P(R)$

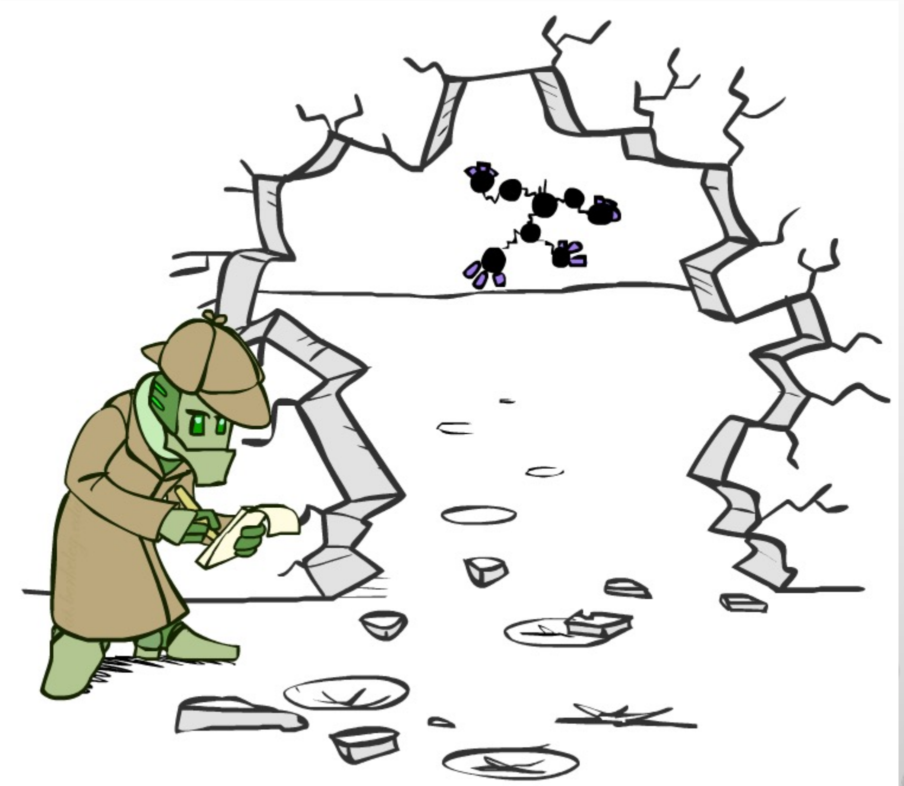
+r	0.1
-r	0.9

$P(T|R)$

+r	+t	0.8
+r	-t	0.2
-r	+t	0.1
-r	-t	0.9

$P(L|T)$

+t	+l	0.3
+t	-l	0.7
-t	+l	0.1
-t	-l	0.9



  - Computing $P(L|+r)$, the initial factors become:

$P(+r)$

+r	0.1

$P(T|+r)$

+r	+t	0.8
+r	-t	0.2

$P(L|T)$

+t	+l	0.3
+t	-l	0.7
-t	+l	0.1
-t	-l	0.9

- We eliminate all vars other than query + evidence

Evidence II

- Result will be a selected joint of query and evidence
  - e.g. For P(L | +r), we would end up with:

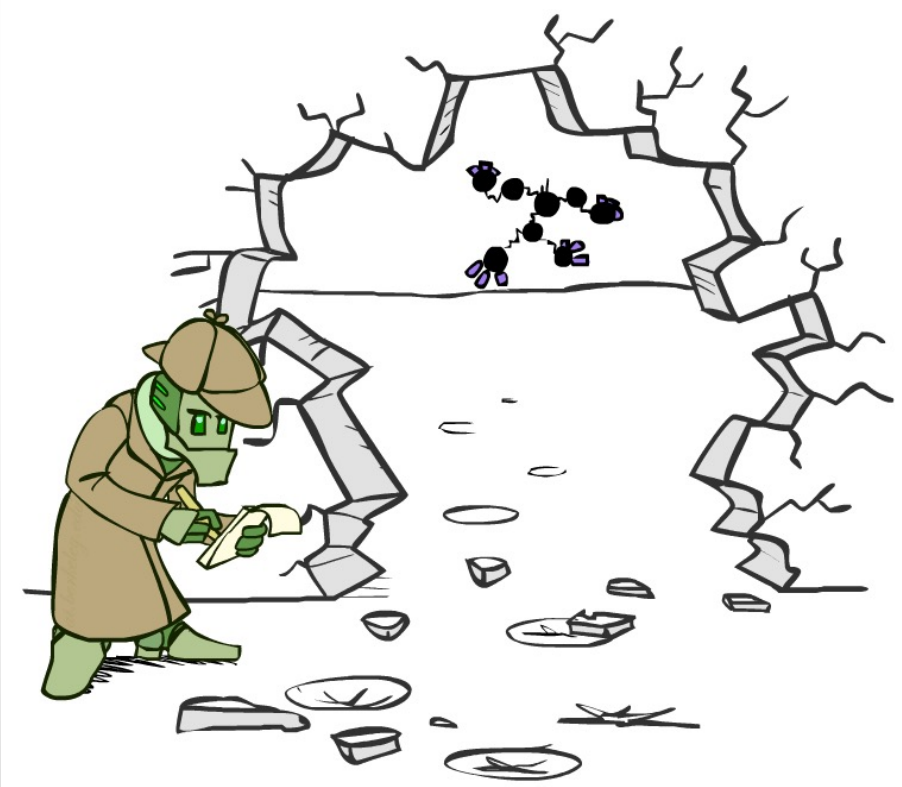
$P(+r, L)$

+r	+l	0.026
+r	-l	0.074

Normalize →

$P(L | +r)$

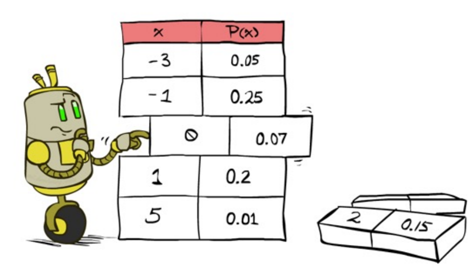
+l	0.26
-l	0.74



- To get our answer, just normalize this!
- That 's it!

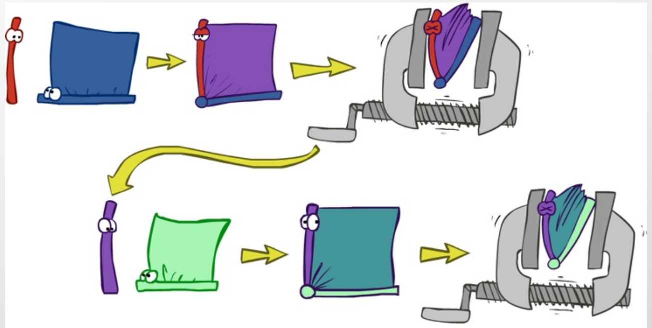
General Variable Elimination

- Query: $P(Q|E_1 = e_1, \ldots E_k = e_k)$



- Start with initial factors:
  - Local CPTs (but instantiated by evidence)

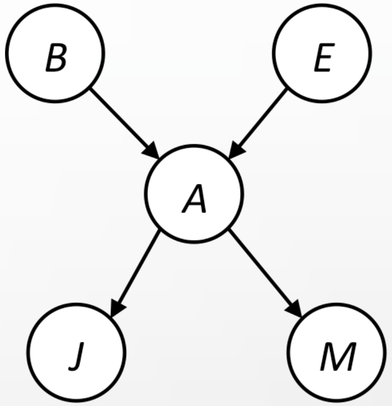
- While there are still hidden variables (not Q or evidence):
  - Pick a hidden variable H
  - Join all factors mentioning H
  - Eliminate (sum out) H



- Join all remaining factors and normalize

$\times \frac{1}{Z}$

Example



$$P(B|j,m) \propto P(B,j,m)$$

$P(B)$	$P(E)$	$P(A|B,E)$	$P(j|A)$	$P(m|A)$

Choose A

$P(A|B,E)$

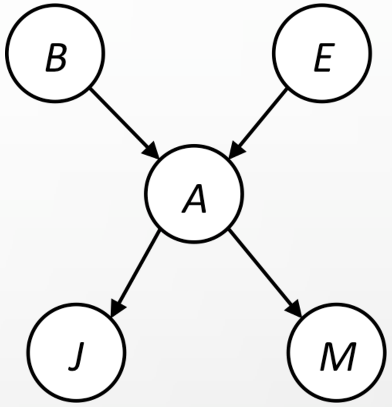
$P(j|A)$ $\xrightarrow{\times}$ $P(j,m,A|B,E)$ $\xrightarrow{\Sigma}$ $P(j,m|B,E)$

$P(m|A)$

$P(B)$	$P(E)$	$P(j,m|B,E)$

Example (cont.)

$$P(B) \qquad P(E) \qquad P(j, m|B, E)$$



Choose E

$P(E)$, $P(j, m|B, E)$ $\xrightarrow{\times}$ $P(j, m, E|B)$ $\xrightarrow{\Sigma}$ $P(j, m|B)$

$$P(B) \qquad P(j, m|B)$$

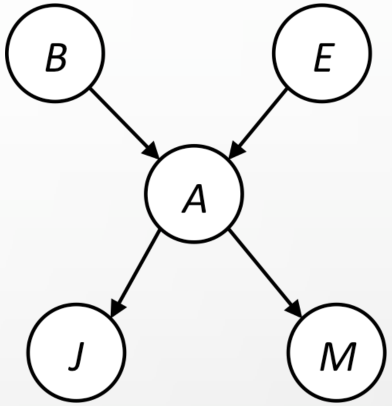
Finish with B

$P(B)$, $P(j, m|B)$ $\xrightarrow{\times}$ $P(j, m, B)$ $\xrightarrow{\text{Normalize}}$ $P(B|j, m)$

Same Example in Equations

$$P(B|j,m) \propto P(B,j,m)$$

$P(B)$	$P(E)$	$P(A|B,E)$	$P(j|A)$	$P(m|A)$



$$\begin{aligned} P(B|j,m) &\propto P(B,j,m) \\ &= \sum_{e,a} P(B,j,m,e,a) \\ &= \sum_{e,a} P(B)P(e)P(a|B,e)P(j|a)P(m|a) \\ &= \sum_{e} P(B)P(e)\sum_{a} P(a|B,e)P(j|a)P(m|a) \\ &= \sum_{e} P(B)P(e)f_1(B,e,j,m) \\ &= P(B)\sum_{e} P(e)f_1(B,e,j,m) \\ &= P(B)f_2(B,j,m) \end{aligned}$$

Marginal can be obtained from joint by summing out

Use Bayes' Net joint distribution expression

Use x*(y+z) = xy + xz

Joining on A, and then summing out gives f_1

Use x*(y+z) = xy + xz

Joining on E, and then summing out gives f_2

All we are doing is exploiting uwy + uwz + uxy + uxz + vwy + vwz + vxy +vxz = (u+v)(w+x)(y+z) to improve computational efficiency!

Another Variable Elimination Example

Query: $P(X_3|Y_1 = y_1, Y_2 = y_2, Y_3 = y_3)$

Start by inserting evidence, which gives the following initial factors:

$$p(Z)p(X_1|Z)p(X_2|Z)p(X_3|Z)p(y_1|X_1)p(y_2|X_2)p(y_3|X_3)$$

Eliminate X_1, this introduces the factor $f_1(Z, y_1) = \sum_{x_1} p(x_1|Z)p(y_1|x_1)$, and we are left with:

$$p(Z)f_1(Z, y_1)p(X_2|Z)p(X_3|Z)p(y_2|X_2)p(y_3|X_3)$$

Eliminate X_2, this introduces the factor $f_2(Z, y_2) = \sum_{x_2} p(x_2|Z)p(y_2|x_2)$, and we are left with:

$$p(Z)f_1(Z, y_1)f_2(Z, y_2)p(X_3|Z)p(y_3|X_3)$$

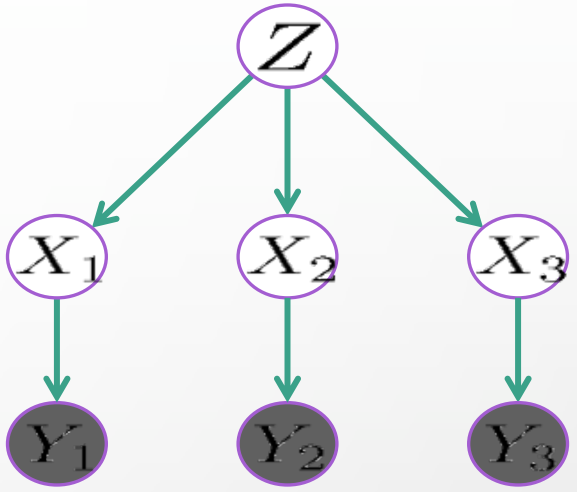
Eliminate Z, this introduces the factor $f_3(y_1, y_2, X_3) = \sum_z p(z)f_1(z, y_1)f_2(z, y_2)p(X_3|z)$, and we are left:

$$p(y_3|X_3), f_3(y_1, y_2, X_3)$$

No hidden variables left. Join the remaining factors to get:

$$f_4(y_1, y_2, y_3, X_3) = P(y_3|X_3)f_3(y_1, y_2, X_3).$$

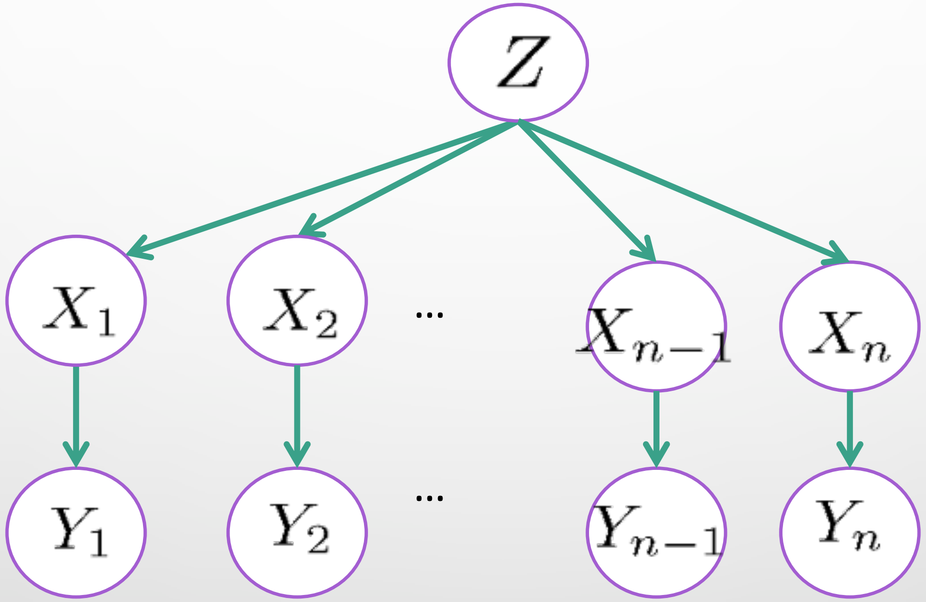
Normalizing over X_3 gives $P(X_3|y_1, y_2, y_3)$.



Computational complexity critically depends on the largest factor being generated in this process. Size of factor = number of entries in table. In example above (assuming binary) all factors generated are of size 2 --- as they all only have one variable (Z, Z, and X_3 respectively).

Variable Elimination Ordering

- For the query $p(X_n | Y_1,\ldots,Y_n)$ work through the following two different orderings as done in previous slide: $Z, X_1, \ldots, X_{n-1}$ and $X_1, \ldots, X_{n-1}, Z$. What is the size of the maximum factor generated for each of the orderings?



- Answer: 2^{n+1} versus 2^2 (assuming binary)

- In general: the ordering can greatly affect efficiency.

VE: Computational and Space Complexity

- The computational and space complexity of variable elimination is determined by the largest factor
- The elimination ordering can greatly affect the size of the largest factor.
  - e.g., Previous slide's example 2^n vs. 2
- Does there always exist an ordering that only results in small factors?
  - No!

Worst Case Complexity?

- CSP:

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (x_2 \vee \neg x_2 \vee x_4) \wedge (\neg x_3 \vee \neg x_4 \vee \neg x_5) \wedge (x_2 \vee x_5 \vee x_7) \wedge (x_4 \vee x_5 \vee x_6) \wedge (\neg x_5 \vee x_6 \vee \neg x_7) \wedge (\neg x_5 \vee \neg x_6 \vee x_7)$$

$P(X_i = 0) = P(X_i = 1) = 0.5$

$Y_1 = X_1 \vee X_2 \vee \neg X_3$

...

$Y_8 = \neg X_5 \vee X_6 \vee X_7$

$Y_{1,2} = Y_1 \wedge Y_2$

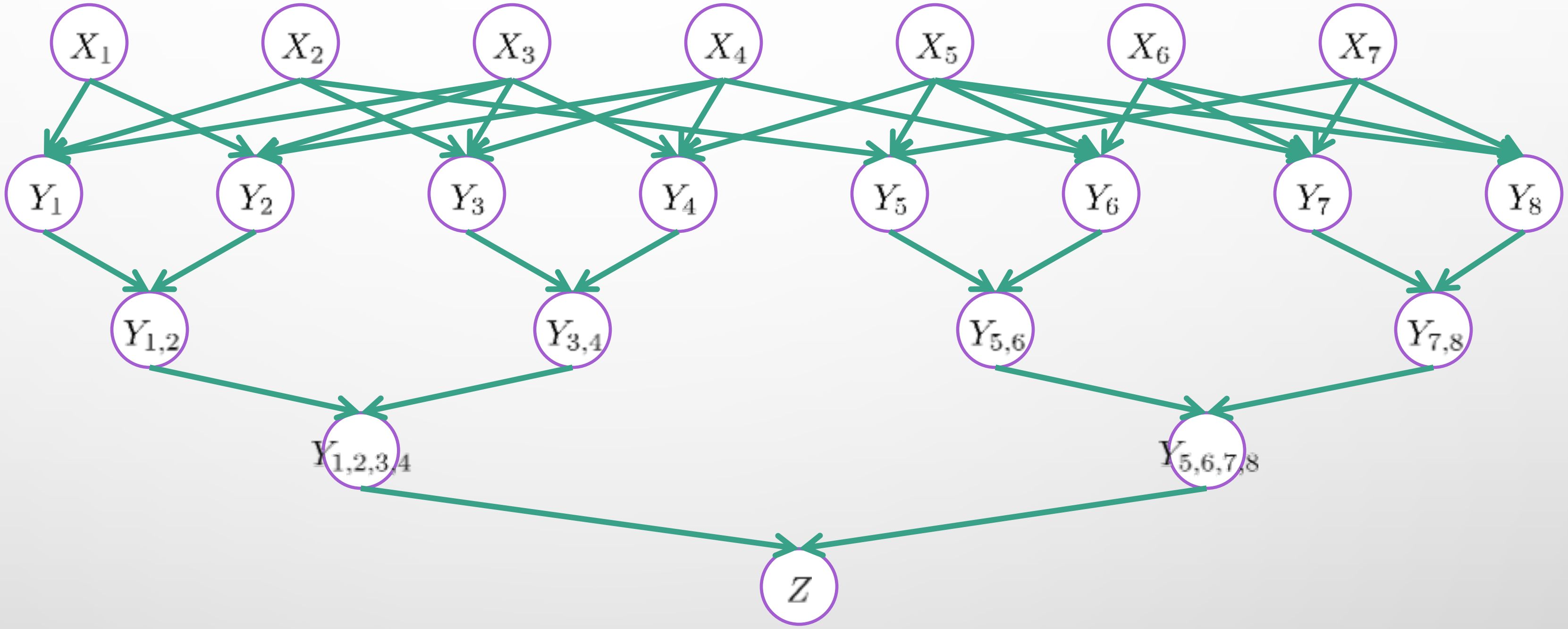
...

$Y_{7,8} = Y_7 \wedge Y_8$

$Y_{1,2,3,4} = Y_{1,2} \wedge Y_{3,4}$

$Y_{5,6,7,8} = Y_{5,6} \wedge Y_{7,8}$

$Z = Y_{1,2,3,4} \wedge Y_{5,6,7,8}$



- If we can answer p(Z = 1) equal to zero or not, we answered whether the 3-SAT problem has a solution.
- Hence inference in Bayes' nets is NP-hard. No known efficient probabilistic inference in general.

Polytrees

- A polytree is a directed graph with no undirected cycles

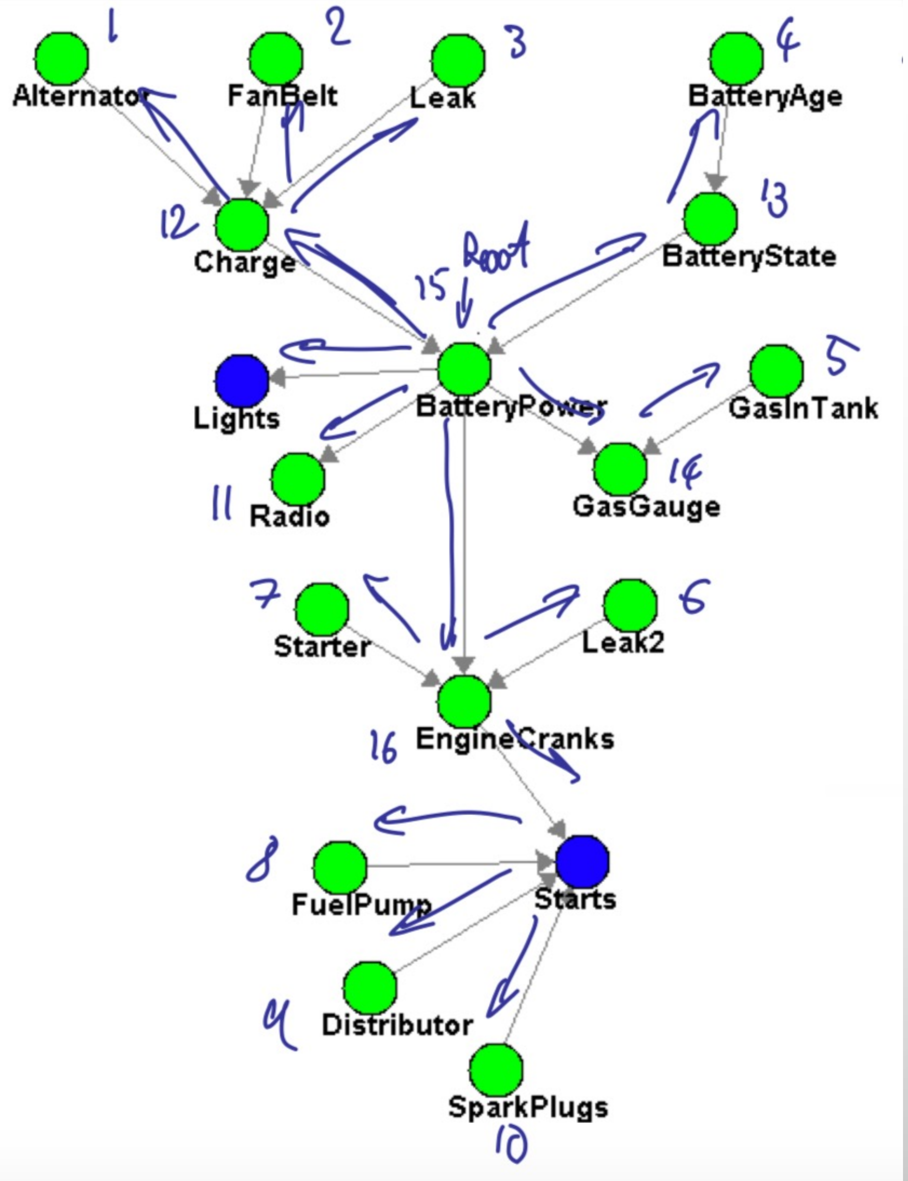
- For poly-trees you can always find an ordering that is efficient
  - Try it!!

- Cut-set conditioning for Bayes' Net inference
  - Choose set of variables such that if removed only a polytree remains
  - Exercise: think about how the specifics would work out!

Variable orders in Polytrees

- Drop edge directions
- Pick some node as a root
- Do a DFS on the root (use undirected edges)
- Eliminate nodes in the reverse topological order of resulting tree.
- Would never get a factor larger than the original CPTs

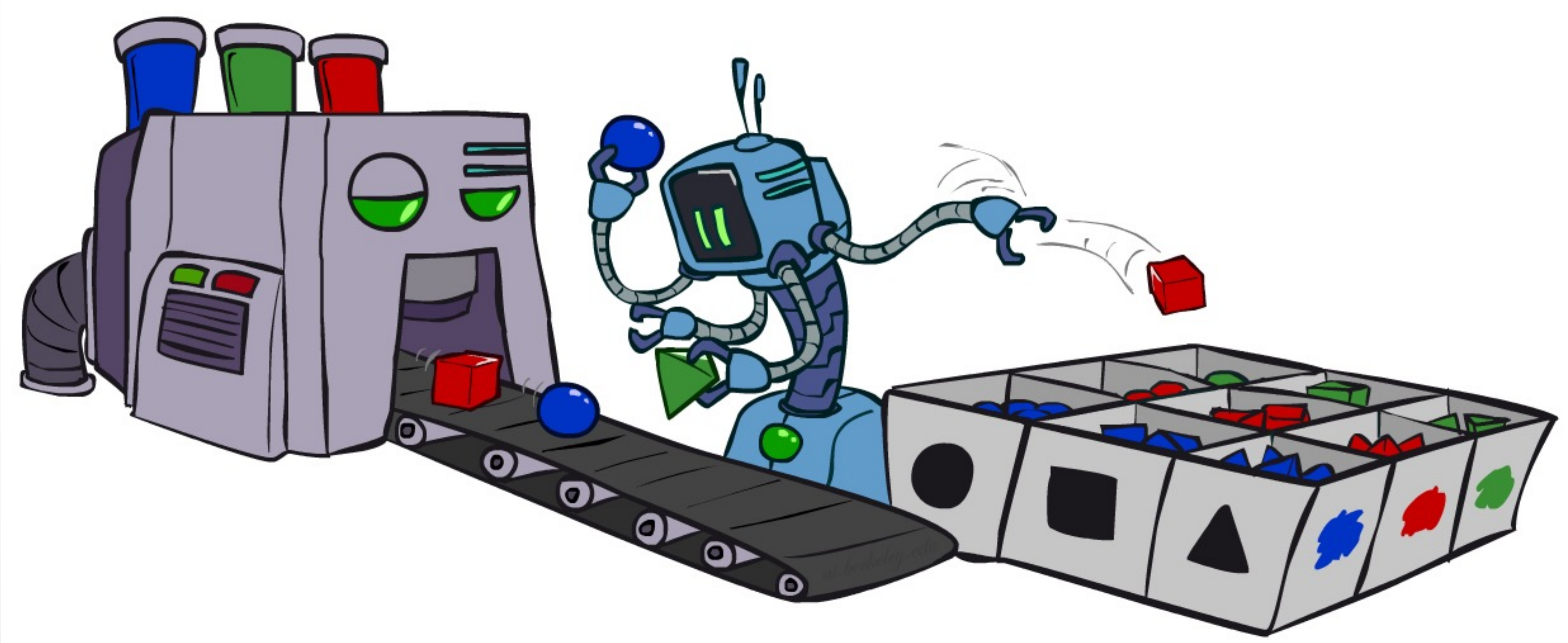
Variable orders in Polytrees (cont.)



Bayes' Nets

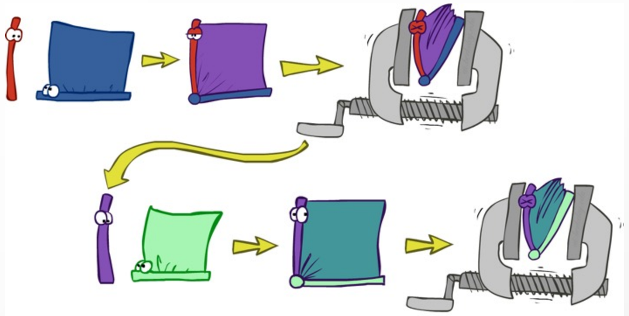
- ✓ Representation
- ✓ Conditional independences
- Probabilistic inference
  - ✓ Enumeration (exact, exponential complexity)
  - ✓ Variable elimination (exact, worst-case exponential complexity, often better)
  - ✓ Inference is np-complete
  - Sampling (approximate)
- Learning bayes' nets from data

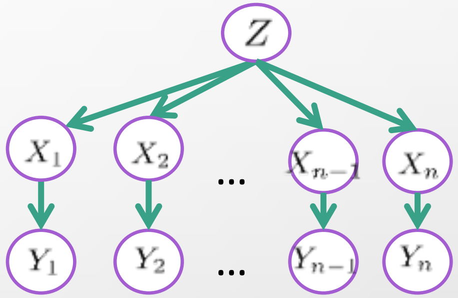
Bayes' Nets: Sampling

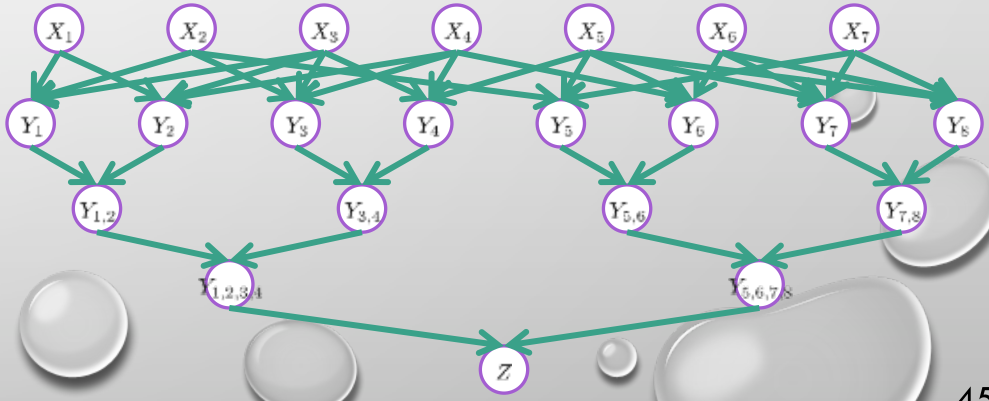


Variable Elimination

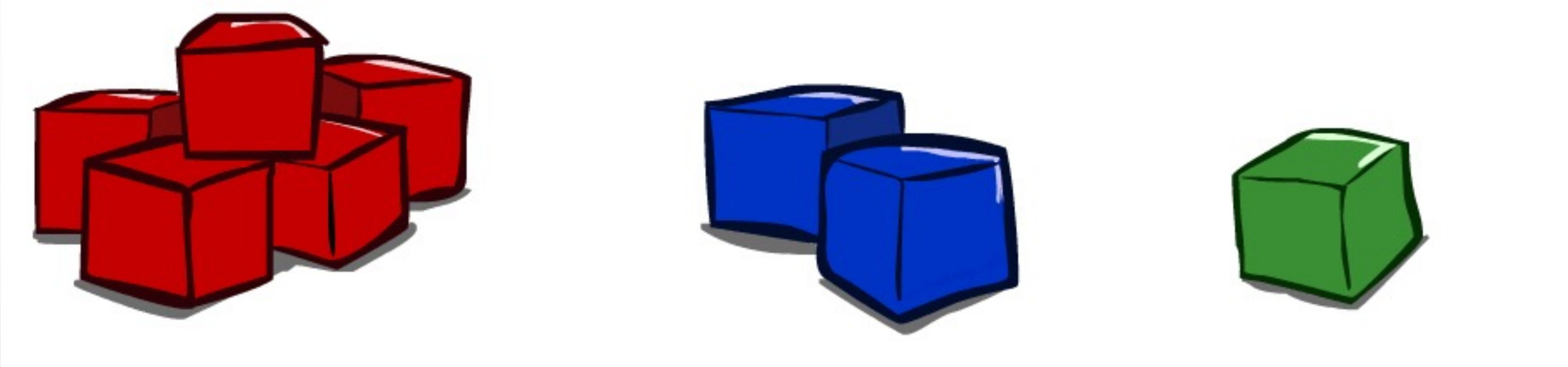
- Interleave joining and marginalizing
- d^k entries computed for a factor over k variables with domain sizes d
- Ordering of elimination of hidden variables can affect size of factors generated
- Worst case: running time exponential in the size of the Bayes' Net







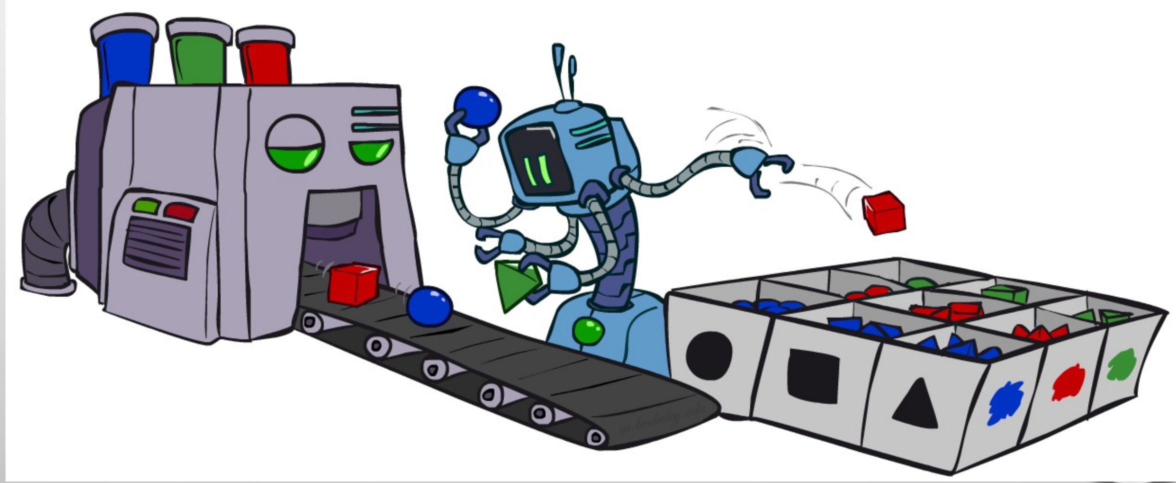
Approximate Inference: Sampling



Sampling

- Sampling is a lot like repeated simulation
  - Predicting the weather, basketball games, …
- Basic idea
  - Draw n samples from a sampling distribution s
  - Compute an approximate posterior probability
  - Show this converges to the true probability p

- Why sample?
  - Learning: get samples from a distribution you don’t know
  - Inference: getting a sample is faster than computing the right answer (e.g. with variable elimination)



Sampling

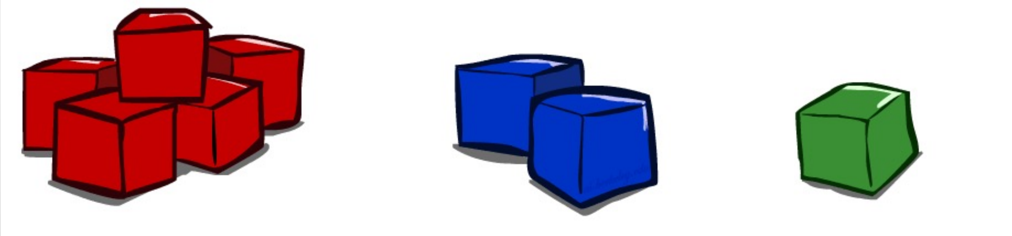
- Sampling from given distribution
  - Step 1: get sample U from uniform distribution over [0, 1)
    - e.g. random() in python
  - Step 2: convert this sample U into an outcome for the given distribution by having each outcome associated with a sub-interval of [0,1) with sub-interval size equal to probability of the outcome

- Example

C	P(C)
red	0.6
green	0.1
blue	0.3

$$0 \leq u < 0.6, \rightarrow C = red$$
$$0.6 \leq u < 0.7, \rightarrow C = green$$
$$0.7 \leq u < 1, \rightarrow C = blue$$

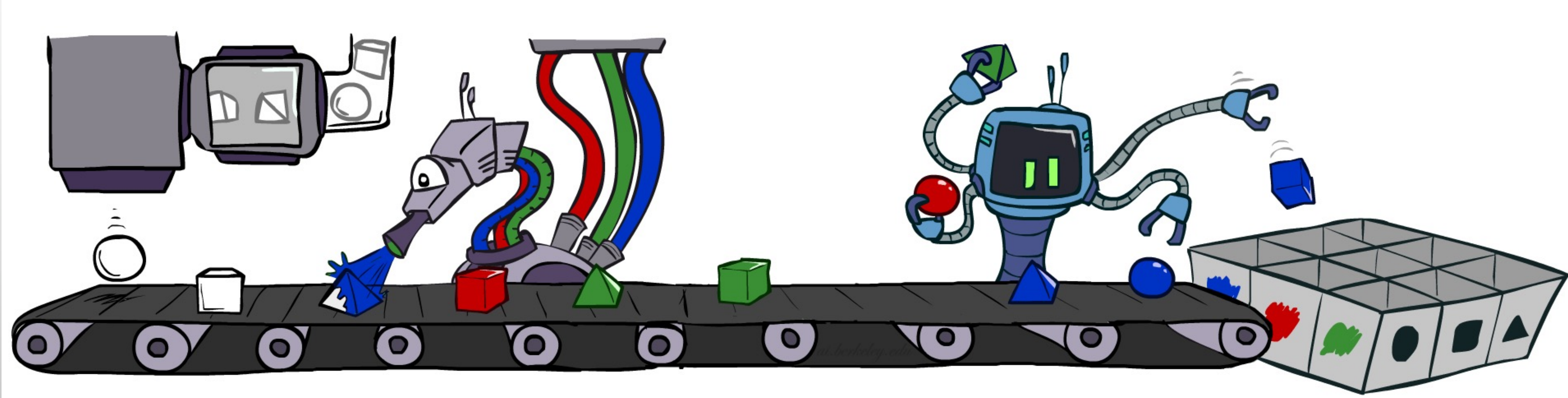
- If random() returns u = 0.83, then our sample is C = blue
- e.g, after sampling 8 times:



Sampling in Bayes' Nets

- Prior sampling
- Rejection sampling
- Likelihood weighting
- Gibbs sampling

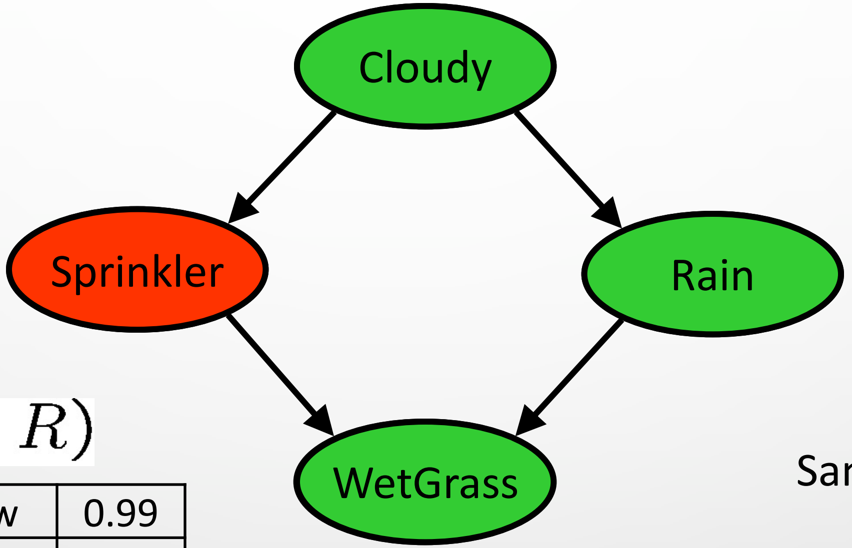
Prior Sampling



Prior Sampling

$P(C)$

+c	0.5
-c	0.5



$P(S|C)$

+c	+s	0.1
	-s	0.9
-c	+s	0.5
	-s	0.5

$P(R|C)$

+c	+r	0.8
	-r	0.2
-c	+r	0.2
	-r	0.8

$P(W|S, R)$

+s	+r	+w	0.99
		-w	0.01
	-r	+w	0.90
		-w	0.10
-s	+r	+w	0.90
		-w	0.10
	-r	+w	0.01
		-w	0.99

Samples:

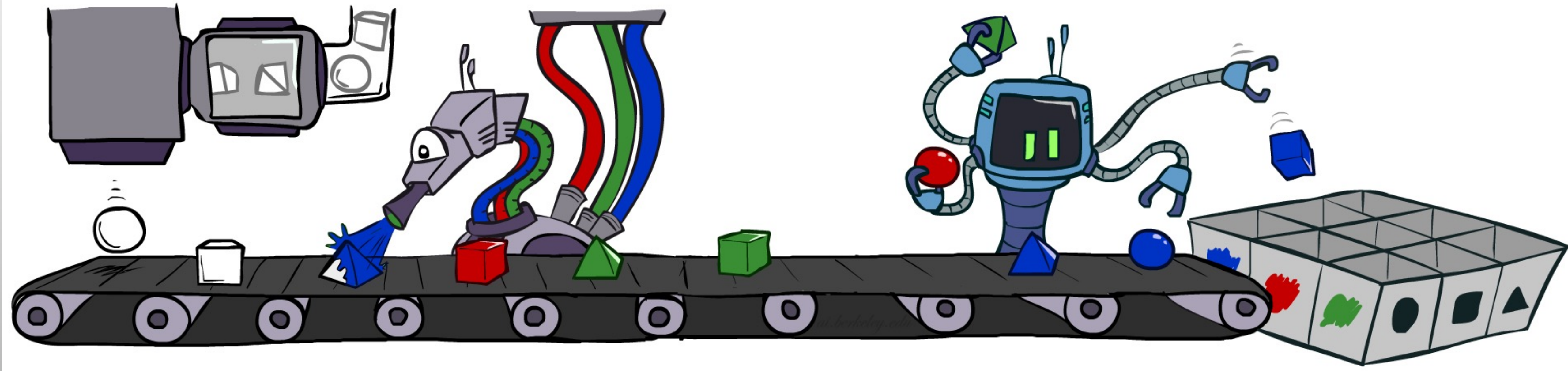
+c, -s, +r, +w

-c, +s, -r, +w

…

Prior Sampling

- For i=1, 2, …, n
  - Sample x_i from $p(X_i \mid parents(X_i))$
- Return $(x_1, x_2, \ldots, x_n)$



Prior Sampling

- This process generates samples with probability:

$$S_{PS}(x_1 \ldots x_n) = \prod_{i=1}^{n} P(x_i|\text{Parents}(X_i)) = P(x_1 \ldots x_n)$$

…i.e. the BNs joint probability

- Let the number of samples of an event be $N_{PS}(x_1 \ldots x_n)$

- Then

$$\begin{aligned} \lim_{N\to\infty} \widehat{P}(x_1, \ldots, x_n) &= \lim_{N\to\infty} N_{PS}(x_1, \ldots, x_n)/N \\ &= S_{PS}(x_1, \ldots, x_n) \\ &= P(x_1 \ldots x_n) \end{aligned}$$

- i.e., The sampling procedure is consistent

Example

- We'll get a bunch of samples from the BN:

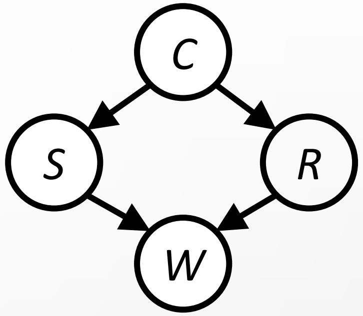
  +c, -s, +r, +w

  +c, +s, +r, +w

  -c, +s, +r, -w

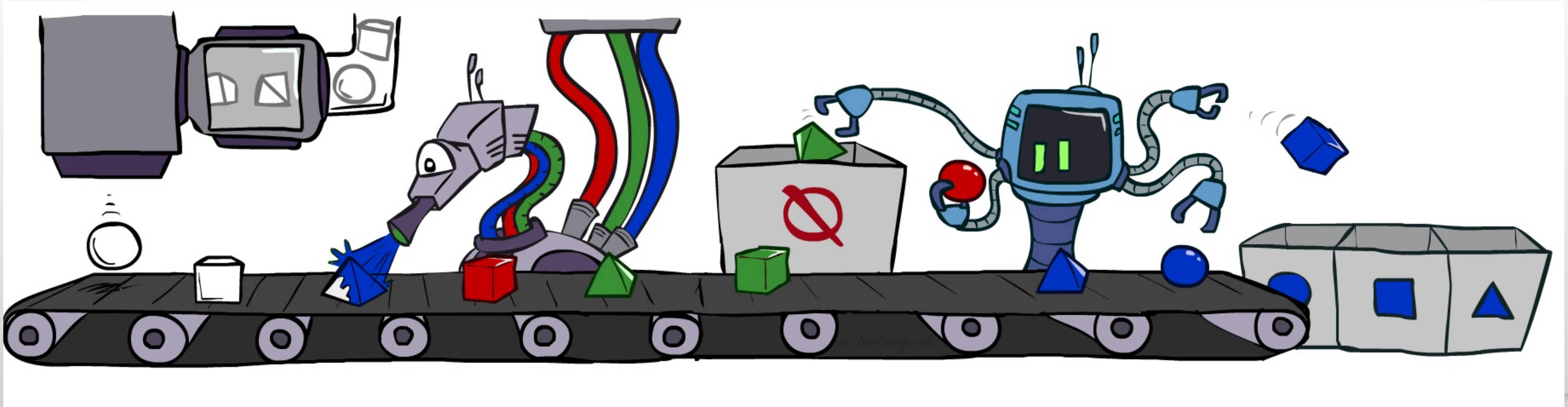
  +c, -s, +r, +w

  -c, -s, -r, +w



- If we want to know P(W)
  - We have counts <+w:4, -w:1>
  - Normalize to get P(W) = <+w:0.8, -w:0.2>
  - This will get closer to the true distribution with more samples
  - Can estimate anything else, too
  - What about P(C| +w)? P(C| +r, +w)? P(C| -r, -w)?
  - Fast: can use fewer samples if less time (what's the drawback?)

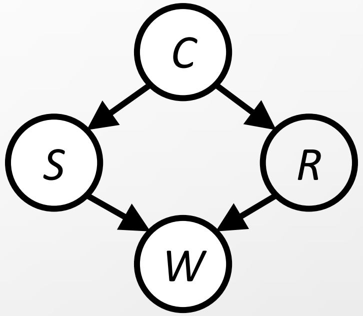
Rejection Sampling



Rejection Sampling

- Let's say we want P(C)
  - No point keeping all samples around
  - Just tally counts of C as we go

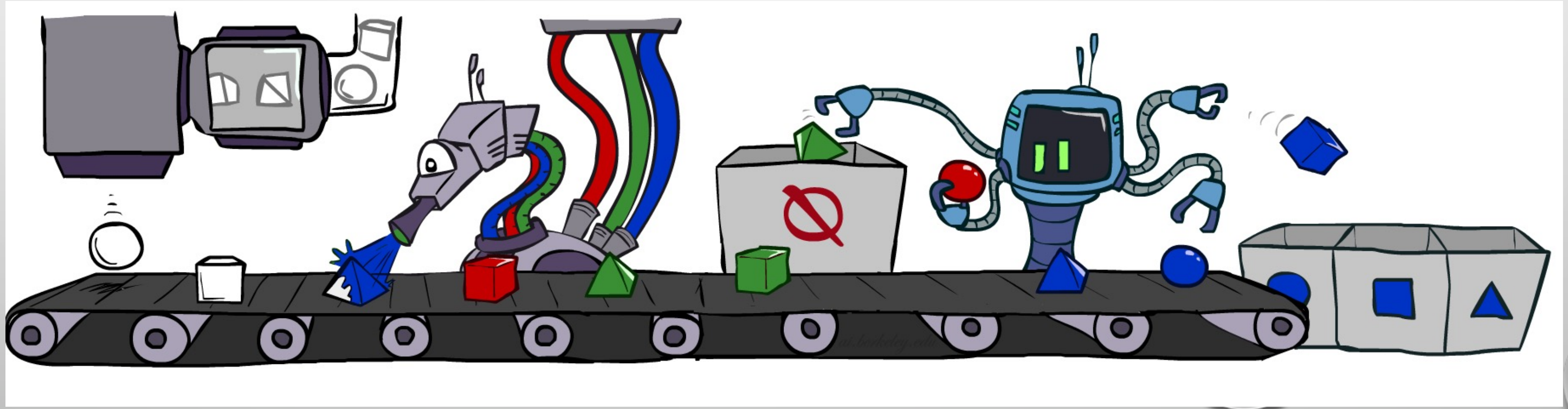
- Let's say we want P(C| +s)
  - Same thing: tally C outcomes, but ignore (reject) samples which don't have S = +s
  - This is called rejection sampling
  - It is also consistent for conditional probabilities (i.e., correct in the limit)



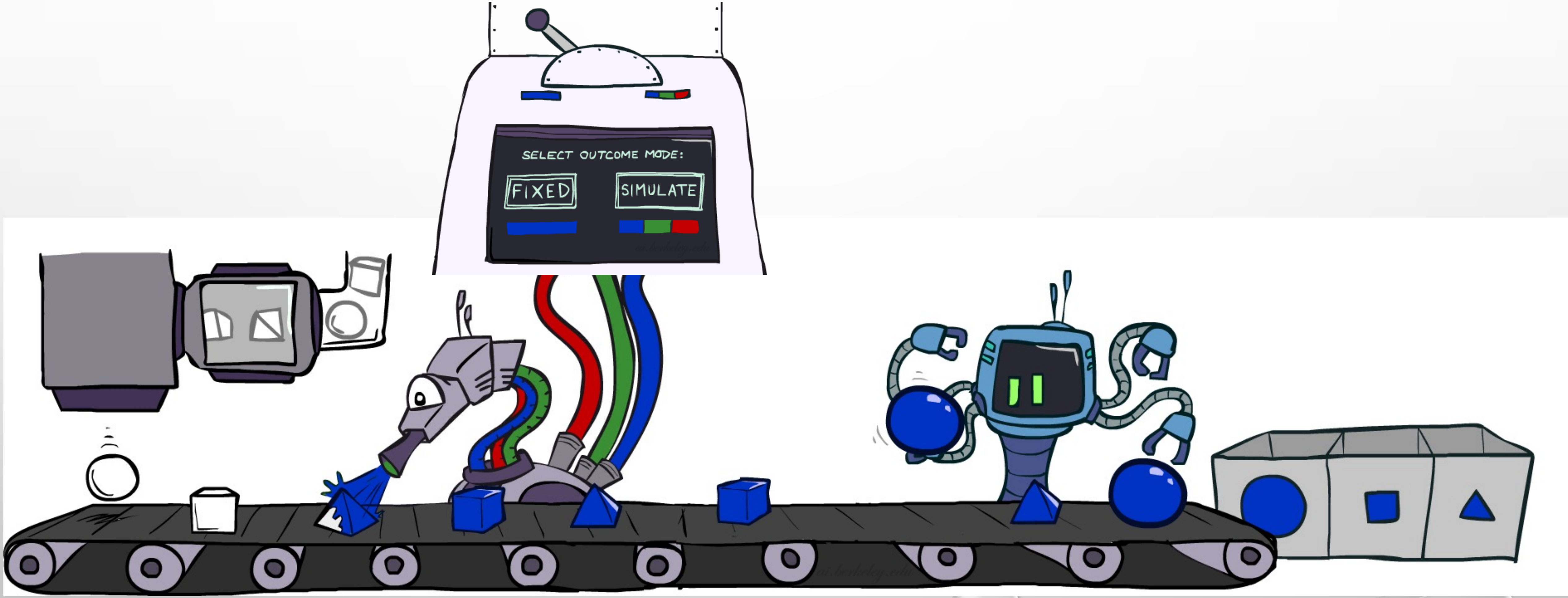
+c, -s, +r, +w
+c, +s, +r, +w
-c, +s, +r, -w
+c, -s, +r, +w
-c, -s, -r, +w

Rejection Sampling

- In: evidence instantiation
- For i=1, 2, …, n
  - Sample x_i from $p(X_i \mid parents(X_i))$
  - If x_i not consistent with evidence
    - Reject: return, and no sample is generated in this cycle
- Return $(x_1, x_2, …, x_n)$

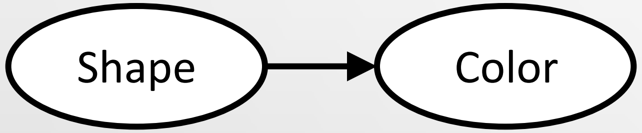


Likelihood Weighting

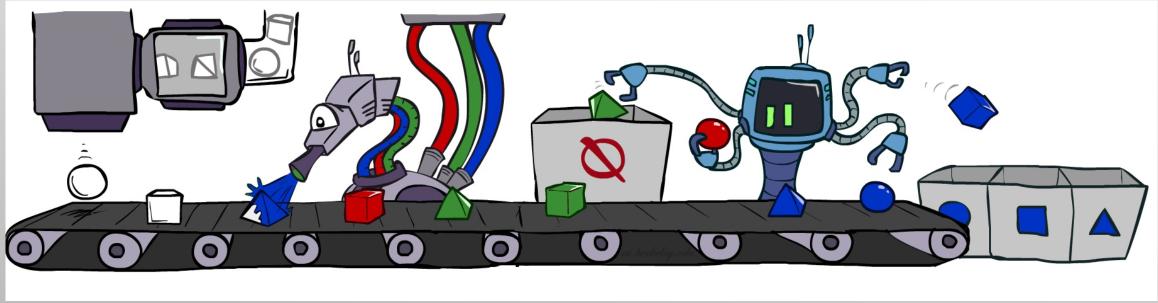


Likelihood Weighting

- Problem with rejection sampling:
  - If evidence is unlikely, rejects lots of samples
  - Evidence not exploited as you sample
  - Consider p(shape | blue)

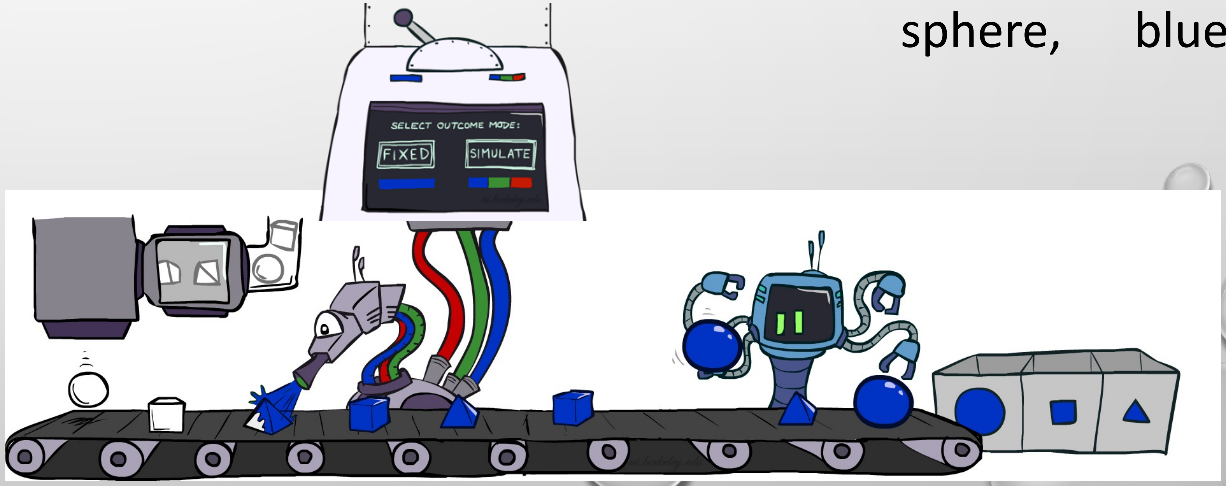


~~pyramid, green~~
~~pyramid, red~~
sphere, blue
~~cube, red~~
~~sphere, green~~



- Idea: fix evidence variables and sample the rest
  - Problem: sample distribution not consistent!
  - Solution: weight by probability of evidence given parents

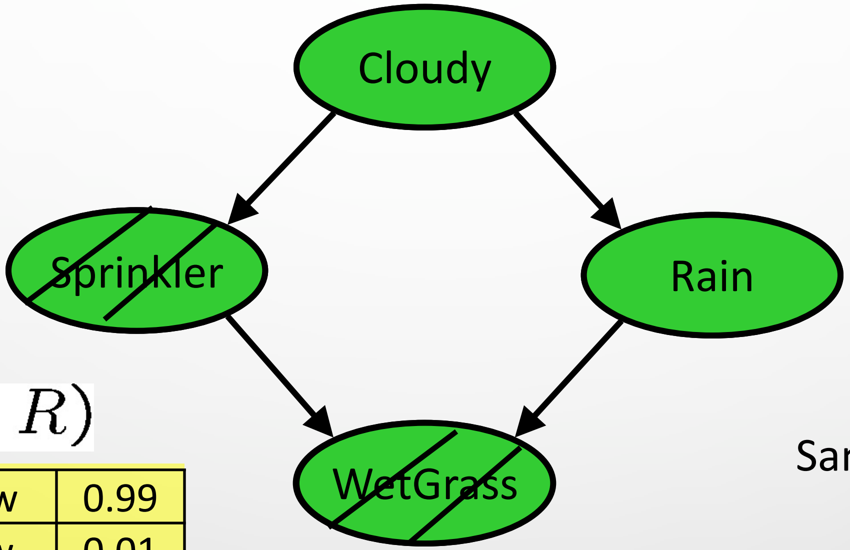
pyramid, blue
pyramid, blue
sphere, blue
cube, blue
sphere, blue



Likelihood Weighting

$P(C)$

+c	0.5
-c	0.5



$P(S|C)$

+c	+s	0.1
	-s	0.9
-c	+s	0.5
	-s	0.5

$P(R|C)$

+c	+r	0.8
	-r	0.2
-c	+r	0.2
	-r	0.8

$P(W|S,R)$

+s	+r	+w	0.99
		-w	0.01
	-r	+w	0.90
		-w	0.10
-s	+r	+w	0.90
		-w	0.10
	-r	+w	0.01
		-w	0.99

Samples:

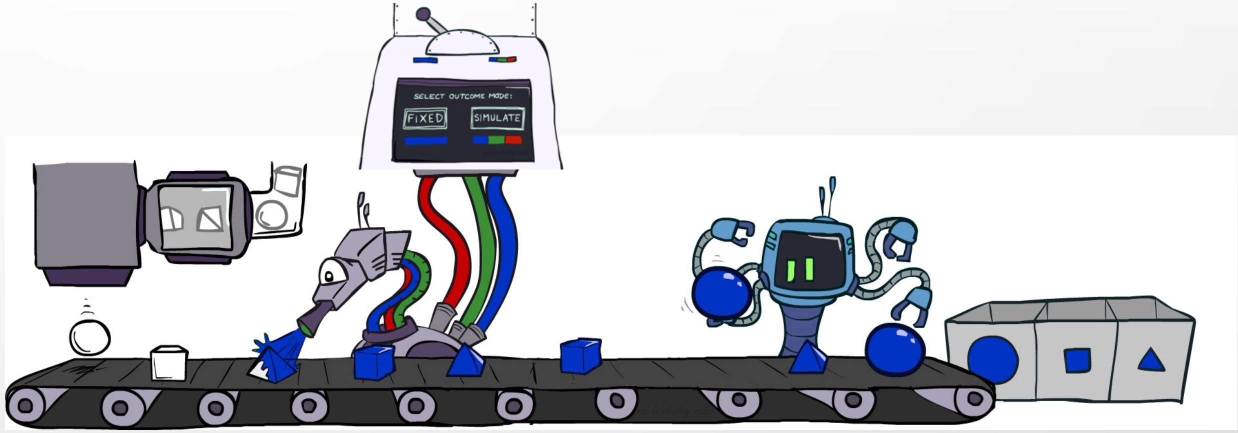
+c, +s, +r, +w

…

$$w = 1.0 \times 0.1 \times 0.99$$

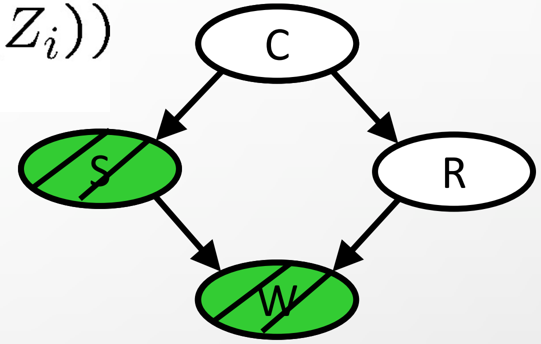
Likelihood Weighting

- In: evidence instantiation
- w = 1.0
- for i=1, 2, …, n
  - if X_i is an evidence variable
    - X_i = observation x_i for X_i
    - Set w = w * p(x_i | parents(X_i))
  - else
    - Sample x_i from p(X_i | parents(X_i))
- Return (x_1, x_2, …, x_n), w



Likelihood Weighting

- Sampling distribution if Z sampled and e fixed evidence $S_{WS}(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^{l} P(z_i|\text{Parents}(Z_i))$



- Now, samples have weights

$$w(\mathbf{z}, \mathbf{e}) = \prod_{i=1}^{m} P(e_i|\text{Parents}(E_i))$$

- Together, weighted sampling distribution is consistent

$$S_{\text{WS}}(z, e) \cdot w(z, e) = \prod_{i=1}^{l} P(z_i|\text{Parents}(z_i)) \prod_{i=1}^{m} P(e_i|\text{Parents}(e_i))$$

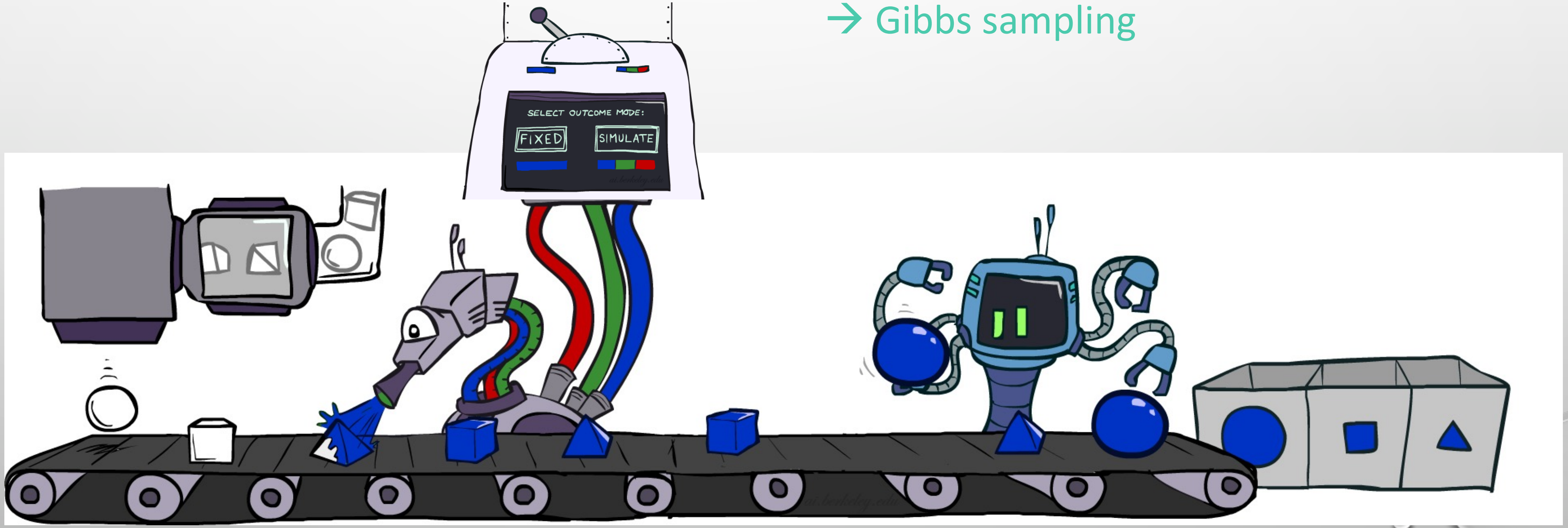
$$= P(\mathbf{z}, \mathbf{e})$$

Likelihood Weighting

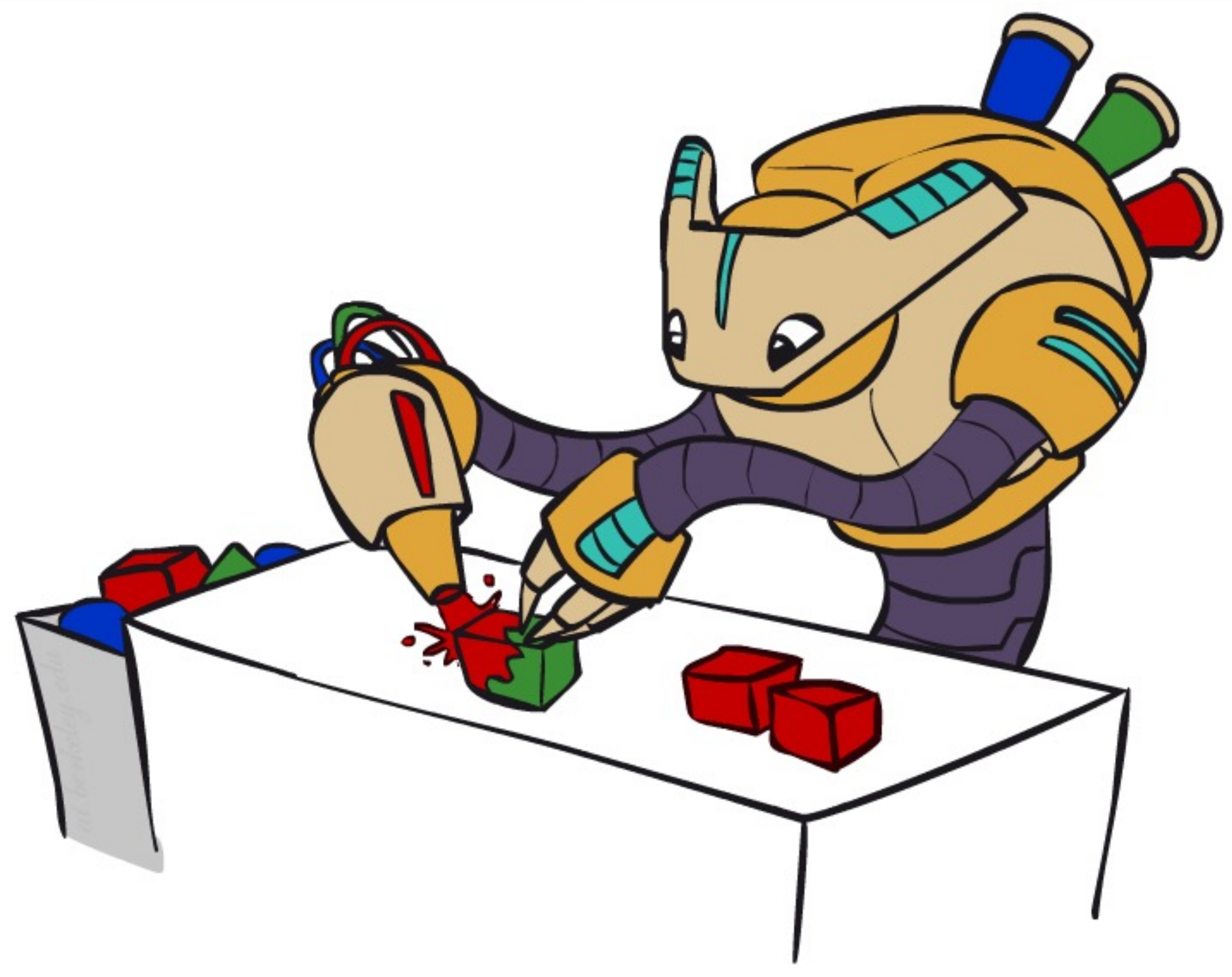
- Likelihood weighting is good
  - We have taken evidence into account as we generate the sample
  - e.g. Here, W's value will get picked based on the evidence values of S, R
  - More of our samples will reflect the state of the world suggested by the evidence

- Likelihood weighting doesn't solve all our problems
  - Evidence influences the choice of downstream variables, but not upstream ones (C isn't more likely to get a value matching the evidence)
- We would like to consider evidence when we sample every variable

  → Gibbs sampling



Gibbs Sampling

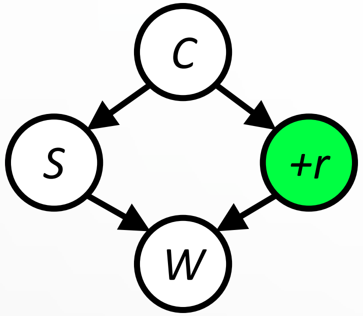


Gibbs Sampling

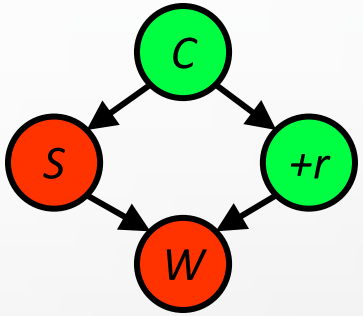
- *Procedure:* keep track of a full instantiation $x_1, x_2, \ldots, x_n$. Start with an arbitrary instantiation consistent with the evidence.
- Sample one variable at a time, **conditioned on *all* the rest**, but keep evidence fixed. Keep repeating this for a long time.
- *Property:* in the limit of repeating this infinitely many times the resulting sample is coming from the correct distribution
- *Rationale*: both upstream and downstream variables condition on evidence.
- In contrast: likelihood weighting only conditions on upstream evidence, and hence weights obtained in likelihood weighting can sometimes be very small. Sum of weights over all samples is indicative of how many "effective" samples were obtained, so want high weight.

Gibbs Sampling Example: P(S | +r)

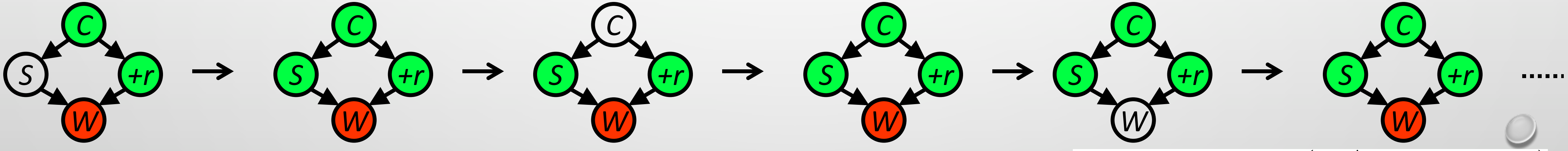
- Step 1: fix evidence
  - R = +r



- Step 2: Initialize other variables
  - Randomly



- Steps 3: repeat
  - Choose a non-evidence variable X
  - Resample X from P(X | all other variables)



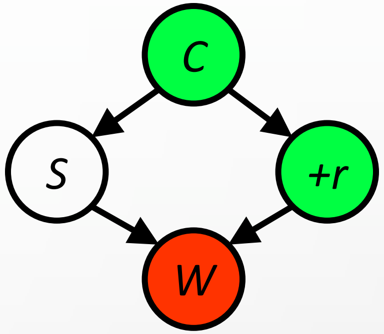
Sample from $P(S|+c,-w,+r)$ Sample from $P(C|+s,-w,+r)$ Sample from $P(W|+s,+c,+r)$

Gibbs Sampling

- How is this better than sampling from the full joint?
  - In a Bayes' Net, sampling a variable given all the other variables (e.g. P(R|S, C, W)) is usually much easier than sampling from the full joint distribution
    - Only requires a join on the variable to be sampled (in this case, a join on R)
    - The resulting factor only depends on the variable's parents, its children, and its children's parents (this is often referred to as its **Markov blanket**)

Efficient Resampling of One Variable

- Sample from P(S | +c, +r, -w)

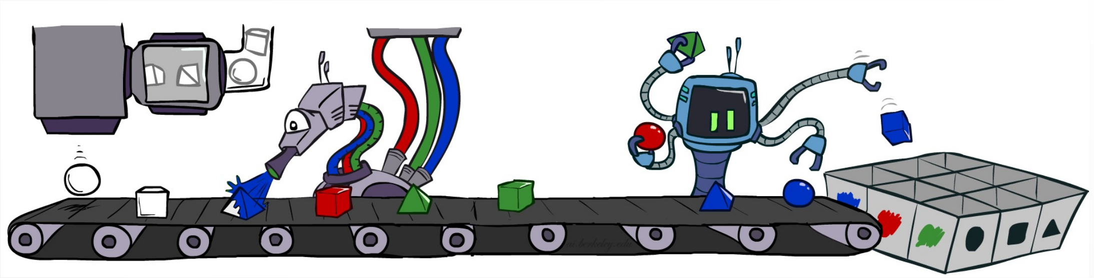


$$
\begin{aligned}
P(S|+c,+r,-w) &= \frac{P(S,+c,+r,-w)}{P(+c,+r,-w)} \\
&= \frac{P(S,+c,+r,-w)}{\sum_s P(s,+c,+r,-w)} \\
&= \frac{P(+c)P(S|+c)P(+r|+c)P(-w|S,+r)}{\sum_s P(+c)P(s|+c)P(+r|+c)P(-w|s,+r)} \\
&= \frac{P(+c)P(S|+c)P(+r|+c)P(-w|S,+r)}{P(+c)P(+r|+c)\sum_s P(s|+c)P(-w|s,+r)} \\
&= \frac{P(S|+c)P(-w|S,+r)}{\sum_s P(s|+c)P(-w|s,+r)}
\end{aligned}
$$

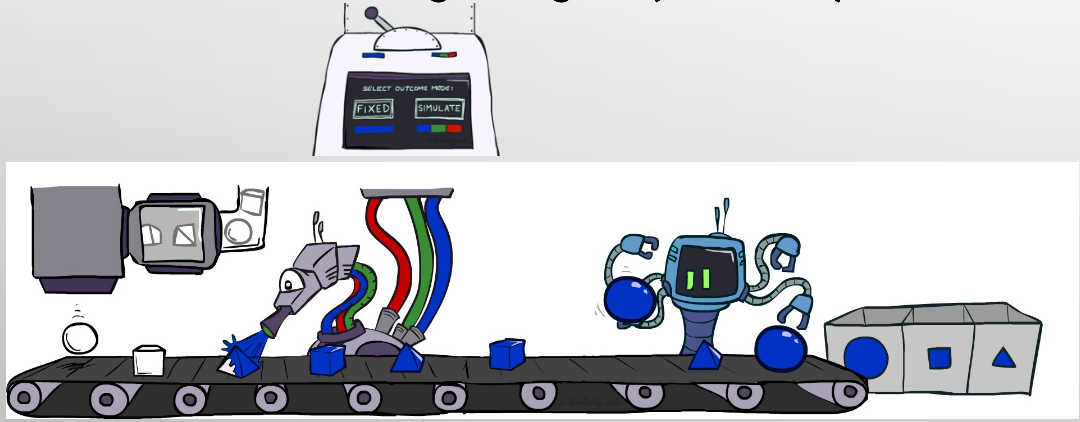
- Many things cancel out – only CPTs with S remain!
- More generally: only CPTs that have resampled variable need to be considered, and joined together

Bayes' Net Sampling Summary

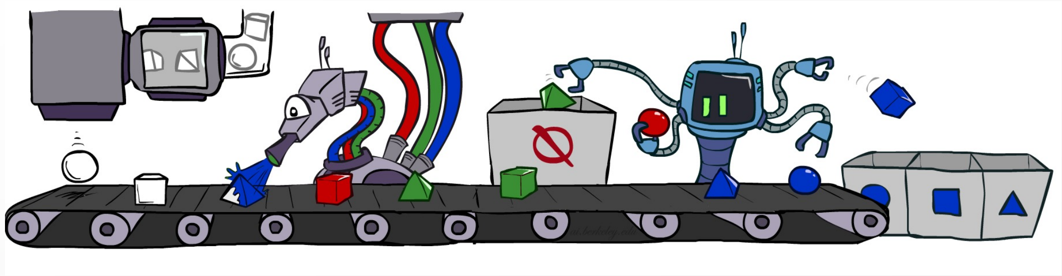
- Prior sampling P



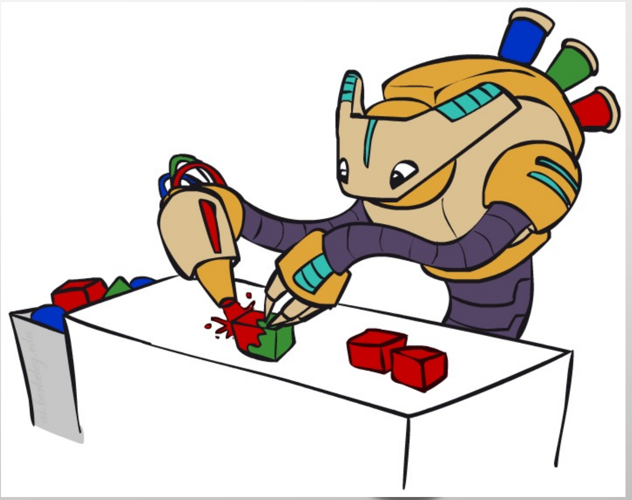
- Likelihood weighting P(Q | e)



- Rejection Sampling P(Q | e)



- Gibbs Sampling P(Q | e)



Further Reading on Gibbs Sampling

- Gibbs sampling produces sample from the query distribution P(Q | e) in limit of re-sampling infinitely often

- Gibbs sampling is a special case of more general methods called Markov Chain Monte Carlo (MCMC) methods

  - Metropolis-Hastings is one of the more famous MCMC methods (in fact, Gibbs sampling is a special case of metropolis-Hastings)

- You may read about Monte Carlo methods – they're just sampling