Bayesian Networks
(and quantum ones)

QCOMICAL School 2026

Claudia Faggian & Gabriele Tedeschi

Quantum theory as a theory of inference

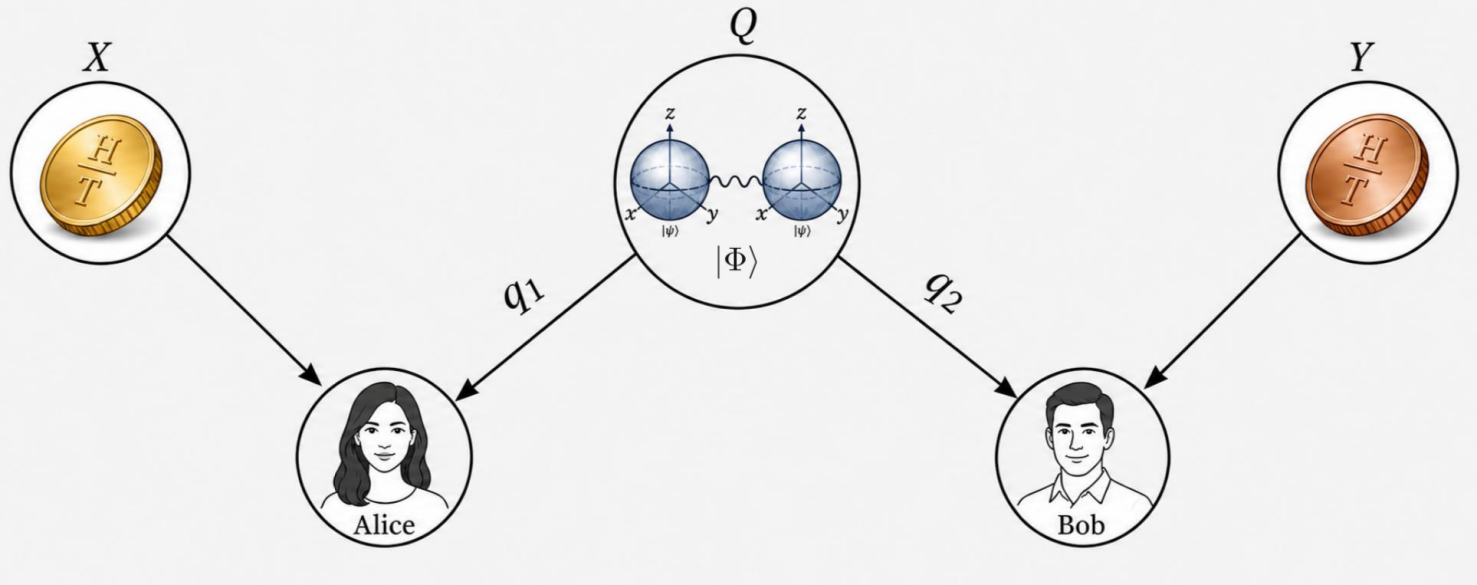
Towards a Formulation of Quantum Theory as a Causally Neutral Theory of Bayesian Inference

Matthew S. Leifer
Chapman University, leifer@chapman.edu

Robert W. Spekkens
Perimeter Institute for Theoretical Physics

- What Quantum Theory does : predict the probabilities of measurement outcomes on a system.
- So framed, prediction is *a problem of probabilistic inference, over models with both classical and quantum data*.
- We already have a theory of probabilistic inference: *Bayesian Inference*.
- **Question:** do tools of classical Bayesian inference carry over to the quantum setting, and how? What more this approach bring ?

Bell set-up



Q = *Preparation of the entangled q_1 and q_2*

X = *Flip a coin*

Alice = *Measure on q_1 parameterized by the value of X*

$$\Pr(a, b \mid x, y) = \Pr(a, b, x, y) / \Pr(x, y)$$

What is *Pr*(*Alice* = 0, *Bob* = 0 | *X* = head, *Y* = tail)?

General Question: Given a system, what is the probability of obtaining a certain measurement?

Plan of this mini-course

Bayesian Networks (classical, and quantum)
from the perspective of programming languages (classical, and quantum)

Part 1:

- Bayesian networks: modelling and inference;
- a probabilistic programming language for **higher-order** models.

Part 2:

- Quantum Bayesian networks (known facts, and research questions)
- a quantum programming language as ongoing work.

Bayesian networks

modelling & inference

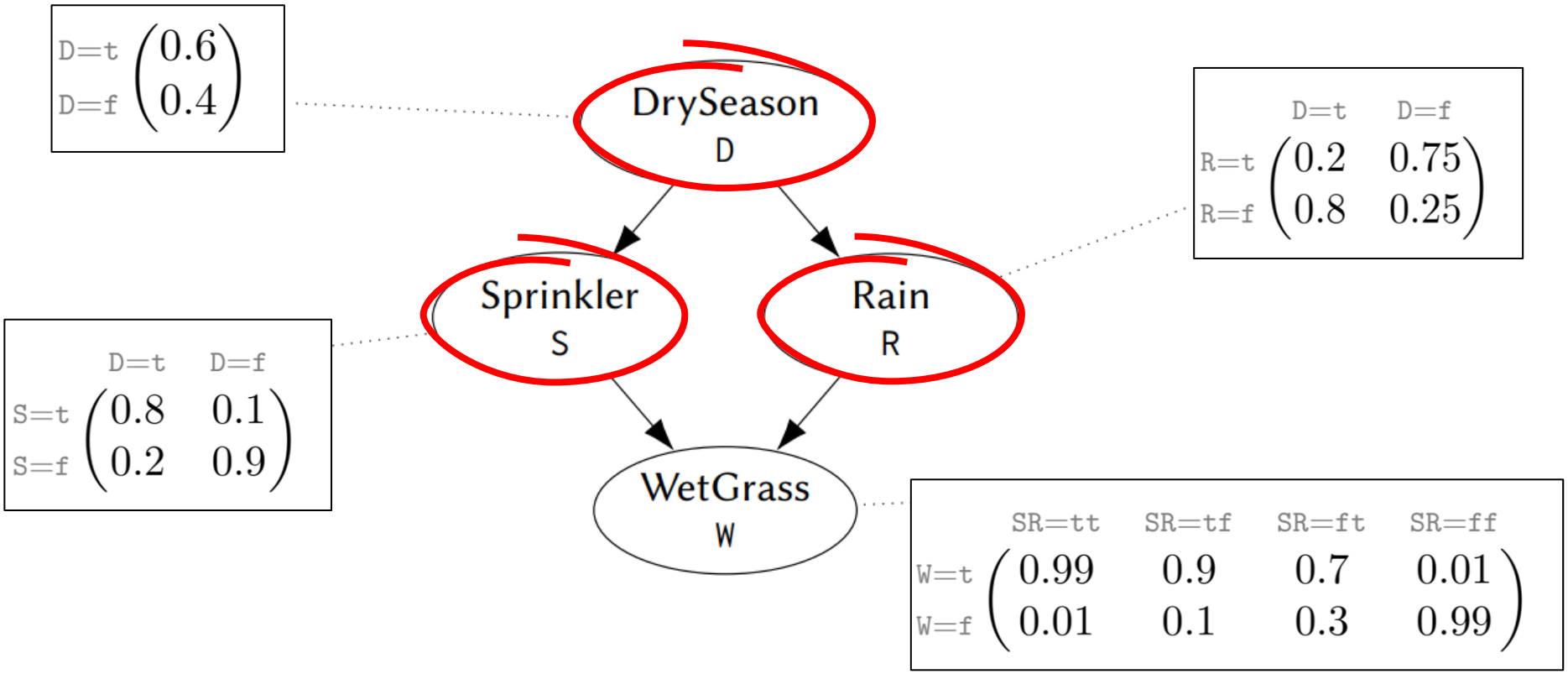
Running example from Darwiche, Modeling and Reasoning with Bayesian Networks (2009)

Bayesian Networks [Pearl 88]

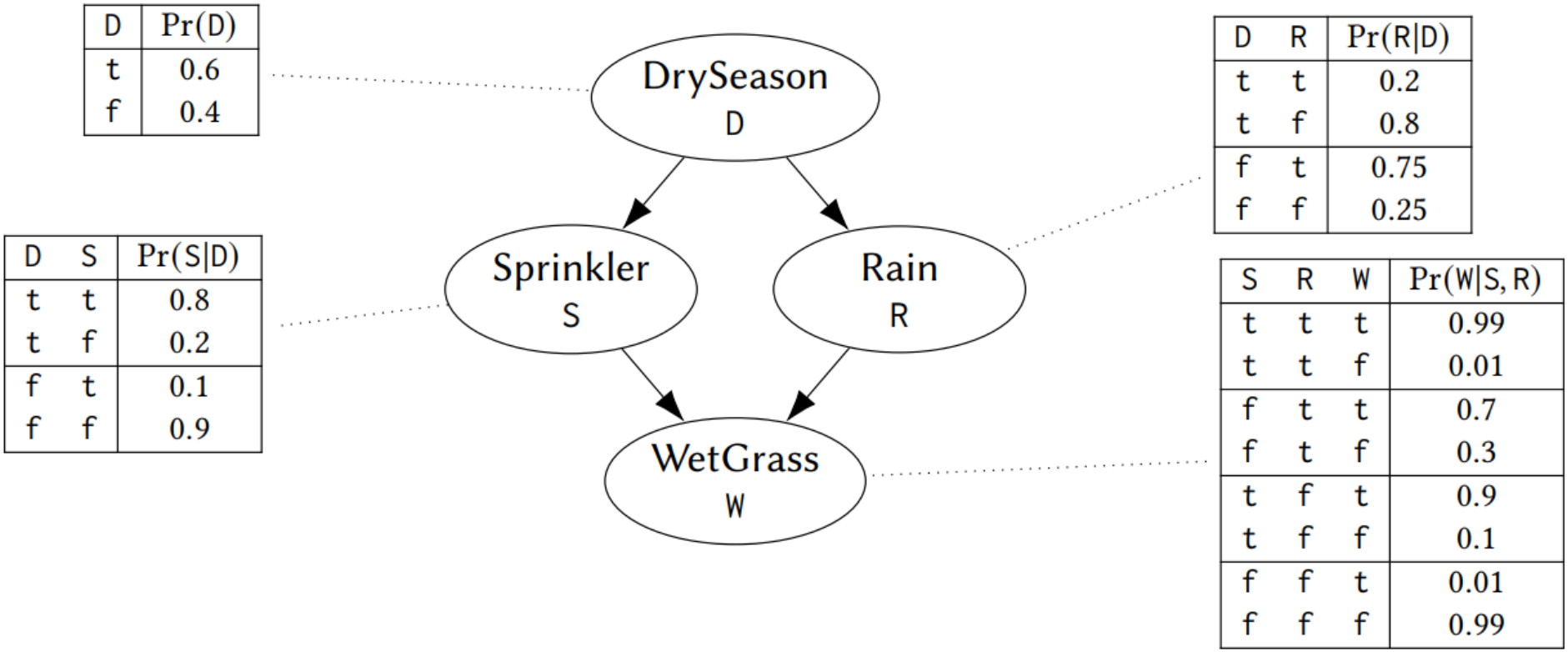
- The strength of **Bayesian networks** [Pearl 88]:
  - represent large probability distributions **compactly**, and
  - availability of **efficient** inference algorithms
    (both **exact** and approximate).

- A Bayesian network consists of two parts:
  1. a qualitative component
     in the form of a directed acyclic graph (DAG)
  2. a quantitative component
     in the form of conditional probabilities.

The fact that the lawn is **Wet** in the morning may depends on either **Rain** or the **Sprinkler** being on. The dependencies between the variables (Rain, Sprinkler, Wet, DrySeason) are pictured in a DAG and their strength is quantified by conditional probability tables (CPT).



The fact that the lawn is **Wet** in the morning may depends on either **Rain** or the **Sprinkler** being on. The dependencies between the variables (Rain, Sprinkler, Wet, DrySeason) are pictured in a DAG and their strength is quantified by conditional probability tables (CPT).



D	Pr(D)
t	0.6
f	0.4

D	S	Pr(S|D)
t	t	0.8
t	f	0.2
f	t	0.1
f	f	0.9

D	R	Pr(R|D)
t	t	0.2
t	f	0.8
f	t	0.75
f	f	0.25

S	R	W	Pr(W|S, R)
t	t	t	0.99
t	t	f	0.01
f	t	t	0.7
f	t	f	0.3
t	f	t	0.9
t	f	f	0.1
f	f	t	0.01
f	f	f	0.99

Bayesian networks, formally

Definition (Bayesian network)

A Bayesian network $\mathcal{B}$ over a finite set of random variables $\mathbf{X}$ is a pair $(\mathcal{G}, \Phi)$ where

- $\mathcal{G}$ is a DAG whose set of nodes is $\mathbf{X}$;
- Φ assigns to each $X \in \mathbf{X}$ a conditional probability table (CPT) ϕ^X for X given its parents.

Theorem (Semantics of BNs) [Pearl 86]

A Bayesian network $\mathcal{B}$ over $\mathbf{X}$ defines a unique probability distribution $\Pr(\mathbf{X})$, given by the product of the CPTs:

$$\Pr(\mathbf{X}) \;=\; \bigodot_{X \in \mathbf{X}} \phi^X \;=\; \prod_{X \in \mathbf{X}} \Pr\big(X \mid \text{parents}(X)\big). \qquad (*)$$

Example: $\Pr(d, s, r, w) \;=\; \Pr(d) \cdot \Pr(s \mid d) \cdot \Pr(r \mid d) \cdot \Pr(w \mid s, r)$

Why factorized: size

The challenge of Bayesian reasoning: a joint distribution is usually too large to represent explicitly.

64 binary variables

joint distribution: $2^{64} \approx 1.8 \cdot 10^{19}$ entries

Bayesian network: 64 CPTs, each of size 2^{k+1} for a variable with k parents

at most 2 parents per variable: $\leq 64 \cdot 2^3 = 512$ entries

With a bound on the number of parents, the network grows linearly in the number of variables, the joint exponentially.

A Bayesian network stores the CPTs only *(factorized representation)*, and inference algorithms answer queries about the joint without ever constructing it *(factorized computation)*.

Q: did it rain last night?

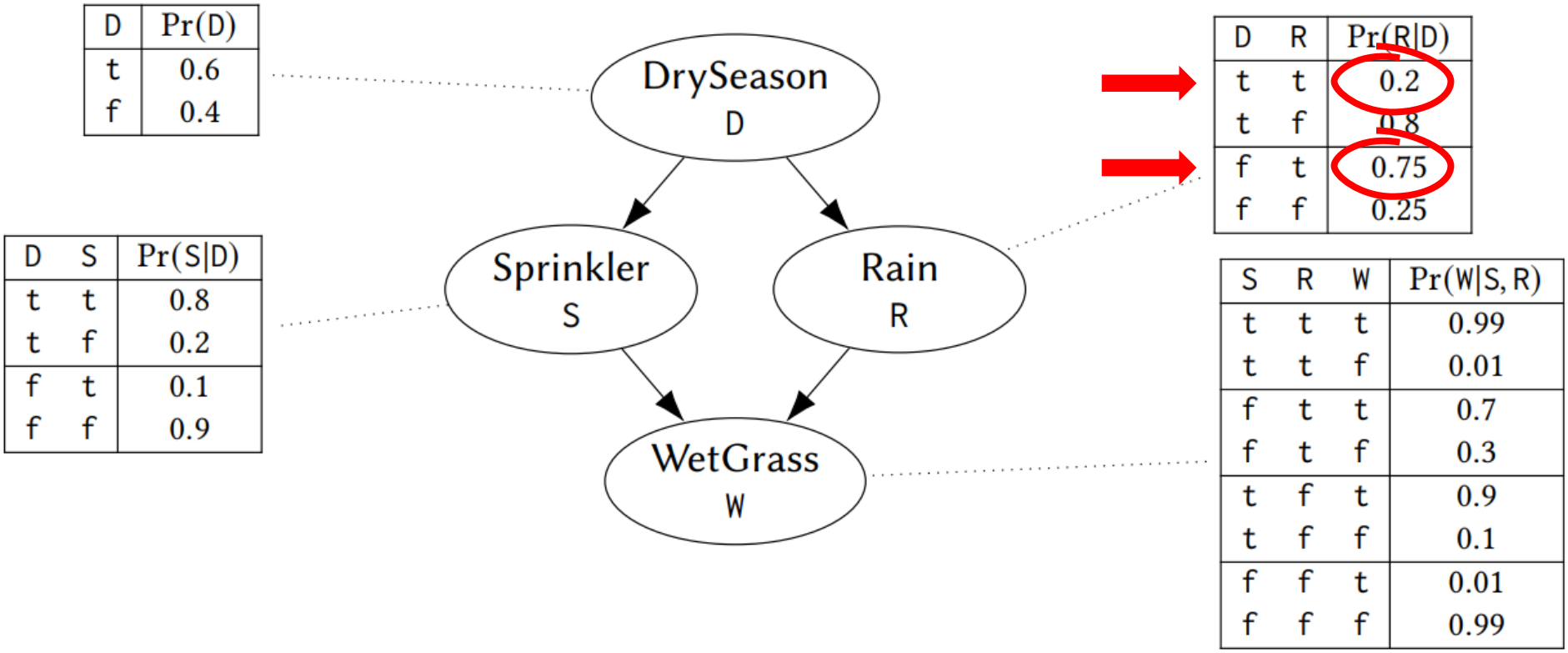
Prior belief: Pr(R = t), we can get is as a **marginal** probability. We observe that the lawn is wet; how does our belief change? Whats the **posterior** probability?

The model lets us compute **marginals** and **posteriors** — both are instances of Bayesian inference, and the posterior is the core of Bayesian conditioning.

Marginals and joints are composition of CPTs

$$Pr(R) = \sum_d Pr\,(R, d) = \sum_d Pr\,(R|d)Pr(d)$$

$$Pr(R = t) = 0.42$$



D	Pr(D)
t	0.6
f	0.4

D	S	Pr(S|D)
t	t	0.8
t	f	0.2
f	t	0.1
f	f	0.9

D	R	Pr(R|D)
t	t	0.2
t	f	0.8
f	t	0.75
f	f	0.25

S	R	W	Pr(W|S, R)
t	t	t	0.99
t	t	f	0.01
f	t	t	0.7
f	t	f	0.3
t	f	t	0.9
t	f	f	0.1
f	f	t	0.01
f	f	f	0.99

Bayesian inference

$$\text{Posterior} = \text{Prior} \times \text{Likelihood} \div \text{Evidence}$$

$$\Pr(\texttt{Rain} = \texttt{t} | \texttt{Wet} = \texttt{t}) = \frac{\Pr(\texttt{Wet} = \texttt{t} | \texttt{Rain} = \texttt{t}) \Pr(\texttt{Rain} = \texttt{t})}{\Pr(\texttt{Wet} = \texttt{t})} = \frac{\Pr(\texttt{Rain} = \texttt{t}, \texttt{Wet} = \texttt{t})}{\Pr(\texttt{Wet} = \texttt{t})}$$

Here Pr(R = t | W = t) = 0.33 / 0.69 = 0.48: the marginals are sums over the joint (left), which is the product of the CPTs.

Joint distribution over D, S, R, W

D	S	R	W	$\Pr(d, s, r, w)$
t	t	t	t	0.09504
t	f	t	t	0.0168
f	t	t	t	0.0297
f	f	t	t	0.189
t	t	t	f	0.00096
t	f	t	f	0.0072
f	t	t	f	0.0003
f	f	t	f	0.081
...				...

$\Longrightarrow$
summing out D, S

Marginal over R, W

R	W	$\Pr(r, w)$
t	t	0.33
t	f	0.09
...		...

$$\Pr(r, w) = \sum_{d,s \in \{\texttt{t},\texttt{f}\}} \Pr(d, s, r, w)$$

Focus on Bayesian **update**

evidence	$\Pr(R = \texttt{t} \mid \text{evidence})$
none (prior)	0.42
$W = \texttt{t}$	0.48
$W = \texttt{t},\ S = \texttt{t}$	0.26
$W = \texttt{t},\ S = \texttt{f}$	0.99

- A wet lawn makes rain more likely: 0.42 → 0.48.
- Learning that the sprinkler was on **explains the lawn away**: 0.48 → 0.26.
- Learning that it was broken makes rain almost certain: 0.48 → **0.99**.

$$\Pr(R \mid e_1, e_2) = \frac{\Pr(R, e_1, e_2)}{\Pr(e_1, e_2)} = \frac{\Pr(e_2 \mid R, e_1)\ \Pr(R \mid e_1)}{\Pr(e_2 \mid e_1)} \qquad \text{posterior after } e_1 = \text{prior for } e_2$$

- Each observation conditions the current belief; **the posterior after one observation is the prior for the next** (sequential update is the same as conditioning on both at once).
- Explaining away: two causes of a common effect, independent given D, become dependent once the effect W is observed. The DAG shows it: S → W ← R.

Dealing with Evidence (factors)

Observing W = t: reject the rows of Pr(W | S, R) which do not agree with the evidence
The result λ(S, R) is a function of S and R, not a CPT.

S	R	W	$\Pr(W \mid S, R)$
t	t	t	0.99
~~t~~	~~t~~	~~f~~	~~0.01~~
t	f	t	0.9
~~t~~	~~f~~	~~f~~	~~0.1~~
f	t	t	0.7
~~f~~	~~t~~	~~f~~	~~0.3~~
f	f	t	0.01
~~f~~	~~f~~	~~f~~	~~0.99~~

$\xrightarrow{W=\mathtt{t}}$

S	R	$\lambda(S, R)$
t	t	0.99
t	f	0.9
f	t	0.7
f	f	0.01

How to organise this computation efficiently: to come.

$$\Pr(R \mid W) = \frac{\Pr(R, W)}{\Pr(W)} \quad \text{where} \quad \Pr(R, W) = \sum_{d,s} \Pr(d)\ \Pr(s \mid d)\ \Pr(R \mid d)\ \Pr(W \mid s, R)$$

$$\Pr(R \mid W = \mathtt{t}) = \frac{\Pr(R, W = \mathtt{t})}{\Pr(W = \mathtt{t})} \quad \text{where} \quad \Pr(R, W = \mathtt{t}) = \sum_{d,s} \Pr(d)\ \Pr(s \mid d)\ \Pr(R \mid d)\ \lambda(s, R)$$

Efficient Inference via Factors

Exact inference algorithms:

Variable elimination
Message passing

Factors

Inference algorithms *rely on two operations on a class of functions,* ***factors****, which generalise probability distributions: joints and CPTs are factors, the partial results of inference are factors (in general, not CPTs), evidence is a factor.*

	R	S	W	$\phi(R,S,W)$
$\mathbf{x}_1$	t	t	t	0.0198
$\mathbf{x}_2$	t	t	f	0.0002
$\mathbf{x}_3$	t	f	t	0.126
$\mathbf{x}_4$	t	f	f	0.054
$\mathbf{x}_5$	f	t	t	0.072
$\mathbf{x}_6$	f	t	f	0.008
$\mathbf{x}_7$	f	f	t	0.0072
$\mathbf{x}_8$	f	f	f	0.7128

Z	X	$\Pr(X \mid Z)$
t	t	0.9
t	f	0.1
f	t	0.2
f	f	0.8

S	R	$\lambda(S,R)$
t	t	0.99
t	f	0.9
f	t	0.7
f	f	0.01

A **factor** ϕ over a finite set of random variables $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a function

$$\phi : \ \mathsf{Val}(\mathbf{X}) = \mathsf{Val}(X_1) \times \cdots \times \mathsf{Val}(X_n) \ \longrightarrow \ \mathbb{R}_0^+$$

mapping each tuple $\mathbf{x}$ of values to a non-negative real.

Sum-out (marginalisation, hiding)

$$\Big(\sum_{Z}\phi\Big)(\mathbf{y}) \triangleq \sum_{z\in \mathsf{Val}(Z)} \phi(\mathbf{y}, z) \qquad \text{for } \phi \text{ over } \mathbf{X} = \mathbf{Y} \uplus \{Z\}$$

Rows that agree on the remaining variables are merged, their values added.

R	W	$\psi_2(R,W)$
t	t	0.3305
t	f	0.0895
f	t	0.3565
f	f	0.2235

$\xrightarrow{\sum_R}$

W	$\Pr(W)$
t	0.687
f	0.313

- **Cost:** linear in the size of the table, O(exp w), w = number of variables in the scope.
- The result is a factor over fewer variables; it is a distribution if the input was one.

Product (composition)

$$(\phi_1 \odot \phi_2)(\mathbf{x}, \mathbf{z}, \mathbf{y}) \triangleq \phi_1(\mathbf{x}, \mathbf{z}) \cdot \phi_2(\mathbf{z}, \mathbf{y}) \qquad \text{for } \phi_1 \text{ over } \mathbf{X} \uplus \mathbf{Z},\ \phi_2 \text{ over } \mathbf{Z} \uplus \mathbf{Y}$$

Z	$\Pr(Z)$
t	0.5
f	0.5

$\odot$

Z	X	$\Pr(X \mid Z)$
t	t	0.9
t	f	0.1
f	t	0.2
f	f	0.8

$\odot$

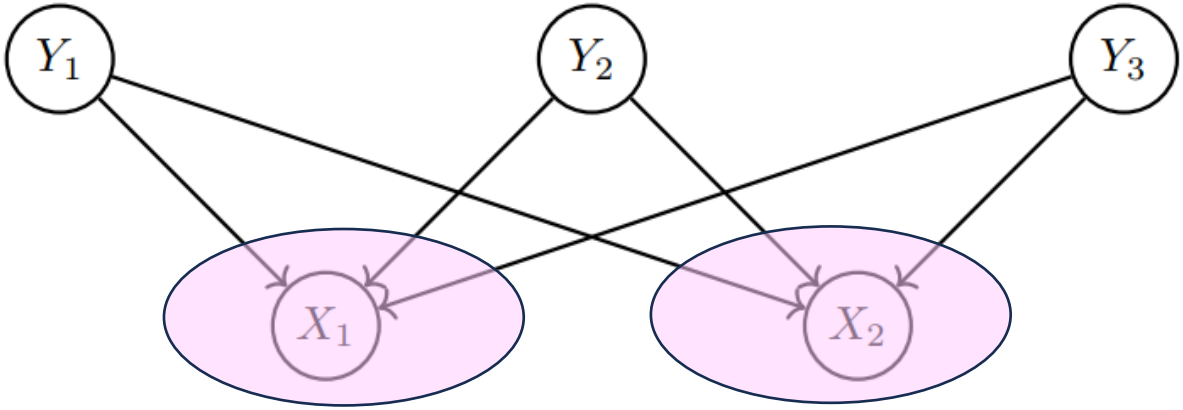
Z	Y	$\Pr(Y \mid Z)$
t	t	0.7
t	f	0.3
f	t	0.4
f	f	0.6

$=$

Z	X	Y	$\Pr(Z, X, Y)$
t	t	t	0.315
t	t	f	0.135
t	f	t	0.035
t	f	f	0.015
f	t	t	0.04
f	t	f	0.06
f	f	t	0.16
f	f	f	0.24

- **Only coherent rows are multiplied:** Pr(Z = t) Pr(X | Z = t) Pr(Y | Z = t), never Pr(Z = t) Pr(X | Z = t) Pr(Y | Z = f). The shared variable Z is read once.
- **Cost:** O(k · exp w) for k factors, w = size of the resulting scope — the product is where blow-up happens.

On the cost of computing the semantics: why **Factors product** is not **Tensor product**



The semantics $[\![X_1]\!]$ of the node X_1 is $\Pr(X_1 \mid Y_1, Y_2, Y_3)$ (a factor $\phi_1(x_1, y_1, y_2, y_3)$), which can be seen as a stochastic matrix of size 2^4. Similarly, the semantics $[\![X_2]\!]$ of the node X_2 is $\Pr(X_2 \mid Y_1, Y_2, Y_3)$ and can be seen as a stochastic matrix with 2^4 entries.

- The tensor product $[\![X_1]\!] \otimes [\![X_2]\!]$ yields a table with 2^8 entries.
- The factors product $[\![X_1]\!] \odot [\![X_2]\!]$ yields a table with 2^5 entries

*In a categorical model, central role is played by **a tensor product** $\otimes$*

Inductive interpretation of a term which encodes a BN over only 5 variables X_1, X_2, Y_1, Y_2, Y_3. may require computing 2^8 values, larger than the full joint!

Factors come with sum-out (marginalization) and product $\odot$

Product and summation are both commutative, product is associative, and—crucially—they distribute under suitable conditions:

$$\text{If } Z \notin \texttt{Nm}(\phi_1) \text{ then } \sum_Z (\phi_1 \odot \phi_2) = \phi_1 \odot (\sum_Z \phi_2)$$

All exact inference rests on distributivity: a sum is pushed inside the product, past the factors that do not mention the variable, so that products stay small.

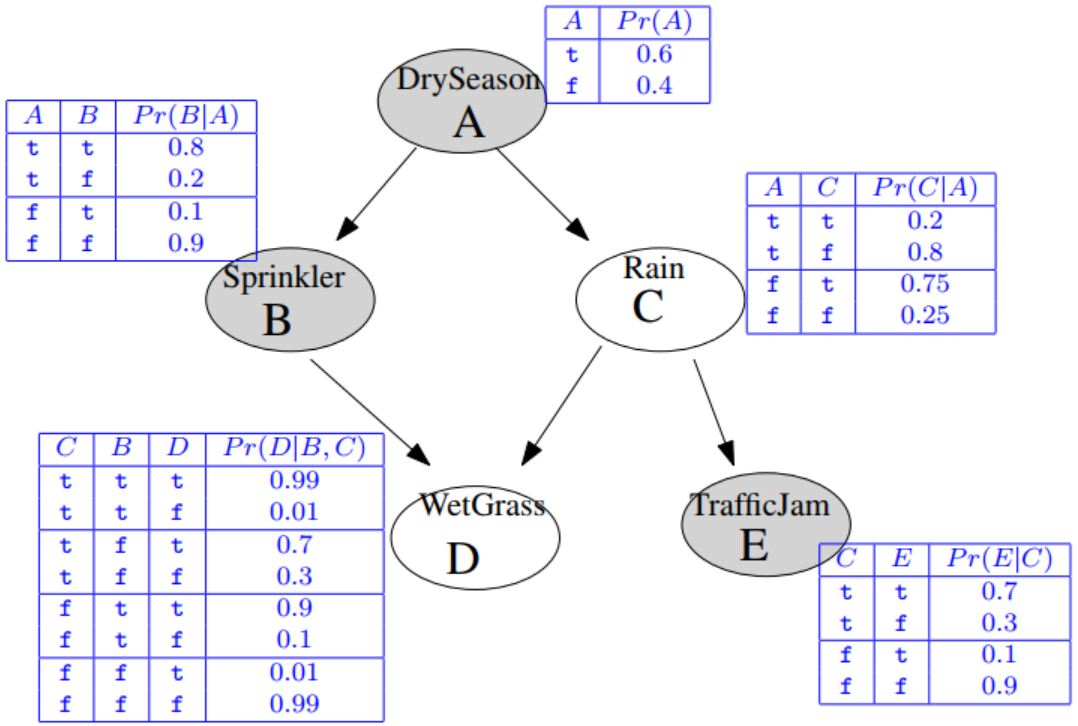
Factorized representation, factorized computation

How to compute efficiently?

Goal: the *marginal distribution over the variables of interest (R,W)*

Inference involves two key operations:

product (*composing*) + summing-out (*hiding*) irrelevant variables.



A	$Pr(A)$
t	0.6
f	0.4

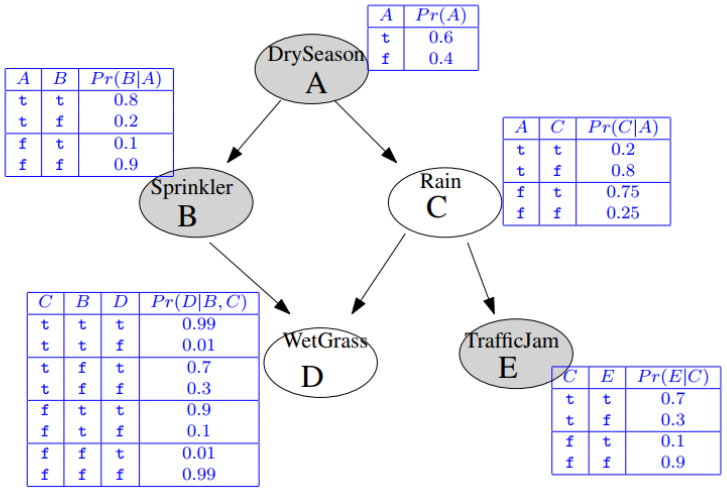
A	B	$Pr(B\|A)$
t	t	0.8
t	f	0.2
f	t	0.1
f	f	0.9

A	C	$Pr(C\|A)$
t	t	0.2
t	f	0.8
f	t	0.75
f	f	0.25

C	B	D	$Pr(D\|B,C)$
t	t	t	0.99
t	t	f	0.01
t	f	t	0.7
t	f	f	0.3
f	t	t	0.9
f	t	f	0.1
f	f	t	0.01
f	f	f	0.99

C	E	$Pr(E\|C)$
t	t	0.7
t	f	0.3
f	t	0.1
f	f	0.9

Variable elimination:
push each sum inside, as far as distributivity allows



A	Pr(A)
t	0.6
f	0.4

A	B	Pr(B|A)
t	t	0.8
t	f	0.2
f	t	0.1
f	f	0.9

A	C	Pr(C|A)
t	t	0.2
t	f	0.8
f	t	0.75
f	f	0.25

C	B	D	Pr(D|B,C)
t	t	t	0.99
t	t	f	0.01
t	f	t	0.7
t	f	f	0.3
f	t	t	0.9
f	t	f	0.1
f	f	t	0.01
f	f	f	0.99

C	E	Pr(E|C)
t	t	0.7
t	f	0.3
f	t	0.1
f	f	0.9

$$\Pr(C,D) = \sum_{A,B,E} \phi^A \, \phi^B \, \phi^C \, \phi^D \, \phi^E$$

$$= \sum_{A,B,E} \Pr(A)\Pr(B \mid A)\Pr(C \mid A)\Pr(D \mid B,C)\Pr(E \mid C)$$

$$= \sum_{E} \Pr(E \mid C) \sum_{B} \Pr(D \mid B,C) \sum_{A} \Pr(A)\Pr(B \mid A)\Pr(C \mid A)$$

Sum out as early as possible

$$\Pr(C, D) = \sum_{A,B,E} \phi^A \, \phi^B \, \phi^C \, \phi^D \, \phi^E = \sum_E \phi^E \sum_B \phi^D \sum_A \phi^A \, \phi^B \, \phi^C$$

Summing out A as early as possible, from the product of the factors containing A, reduces the size of the factors to multiply:

$$\phi(b, c) = \sum_a \Big(\Pr(a) \cdot \Pr(b \mid a) \cdot \Pr(c \mid a) \Big), \tag{1}$$

$$\Pr(c, d) = \sum_e \Big(\Big(\sum_b \phi(b, c) \cdot \Pr(d \mid b, c) \Big) \cdot \Pr(e \mid c) \Big). \tag{2}$$

The successive results are factors over B, C, then C, D, and finally the marginal over C, D. The largest product involves three variables: A, B, C, then B, C, D, then C, D, E. Each such product has 2^3 entries, whereas the full joint distribution has 2^5.

computations	resulting factor	names	cost
$\sum_A \phi^A \odot \phi^B \odot \phi^C$	$\phi^1(B, C)$	ABC	2^3
$\sum_B \phi^D \odot \phi^1(B, C)$	$\phi^2(C, D)$	BCD	2^3
$\sum_E \phi^E \odot \phi^2(C, D)$	$\phi^3(C, D)$	CDE	2^3

The names column lists the variables active at each step, i.e. the scope of the product being computed; the cost is the size of that product.

Step 1: eliminate A

Factors mentioning A: ϕ^A, ϕ^B, ϕ^C. Multiply (over A, B, C), then sum A out: the resulting factor $\phi^1(B, C)$ has 2^2 entries.

A	ϕ^A
t	0.6
f	0.4

A	B	ϕ^B
t	t	0.8
t	f	0.2
f	t	0.1
f	f	0.9

A	C	ϕ^C
t	t	0.2
t	f	0.8
f	t	0.75
f	f	0.25

$\xrightarrow{\odot}$

A	B	C	$\phi^A \odot \phi^B \odot \phi^C$
t	t	t	0.096
t	t	f	0.384
t	f	t	0.024
t	f	f	0.096
f	t	t	0.03
f	t	f	0.01
f	f	t	0.27
f	f	f	0.09

$\xrightarrow{\sum_A}$

B	C	ϕ^1
t	t	0.126
t	f	0.394
f	t	0.294
f	f	0.186

Pool after step 1: $\phi^1(B, C)$, ϕ^D, ϕ^E. ϕ^1 = Pr(B, C): a distribution, but not a CPT of the network.

Step 2: eliminate B

Factors mentioning B: **ϕ^D, $\phi^1(B, C)$**. Multiply (over B, C, D), then sum B out: $\phi^2(C, D)$, 2^2 entries.

B	C	D	ϕ^D
t	t	t	0.99
t	t	f	0.01
t	f	t	0.9
t	f	f	0.1
f	t	t	0.7
f	t	f	0.3
f	f	t	0.01
f	f	f	0.99

$\odot$

B	C	ϕ^1
t	t	0.126
t	f	0.394
f	t	0.294
f	f	0.186

$\xrightarrow{\sum_B}$

C	D	ϕ^2
t	t	0.3305
t	f	0.0895
f	t	0.3565
f	f	0.2235

Pool after step 2: $\phi^2(C, D) = Pr(C, D)$, ϕ^E.

Step 3: eliminate E, read off Pr(C, D)

C	E	ϕ^E
t	t	0.7
t	f	0.3
f	t	0.1
f	f	0.9

$\odot$

C	D	ϕ^2
t	t	0.3305
t	f	0.0895
f	t	0.3565
f	f	0.2235

$\xrightarrow{\sum_E}$

C	D	$\phi^3 = \Pr(C, D)$
t	t	0.3305
t	f	0.0895
f	t	0.3565
f	f	0.2235

computations	resulting factor	names	cost
$\sum_A \phi^A \odot \phi^B \odot \phi^C$	$\phi^1(B, C)$	ABC	2^3
$\sum_B \phi^D \odot \phi^1(B, C)$	$\phi^2(C, D)$	BCD	2^3
$\sum_E \phi^E \odot \phi^2(C, D)$	$\phi^3(C, D)$	CDE	2^3

Three steps, three resulting factors, **each over 2 variables: 2^2 entries.** The joint has 2^5.

$\Sigma_E \phi^E = 1$: E is barren for this query and could have been pruned.

With evidence: Pr(C | D = t)

Replace ϕ^D = Pr(D | B, C) by its reduction λ(B, C) = Pr(D = t | B, C). Same order A, B, E: step 1 is unchanged (ϕ^1), step 2 multiplies by λ instead of ϕ^D.

B	C	$\lambda(B,C)$
t	t	0.99
t	f	0.9
f	t	0.7
f	f	0.01

$\odot$

B	C	ϕ^1
t	t	0.126
t	f	0.394
f	t	0.294
f	f	0.186

$\xrightarrow{\sum_B}$

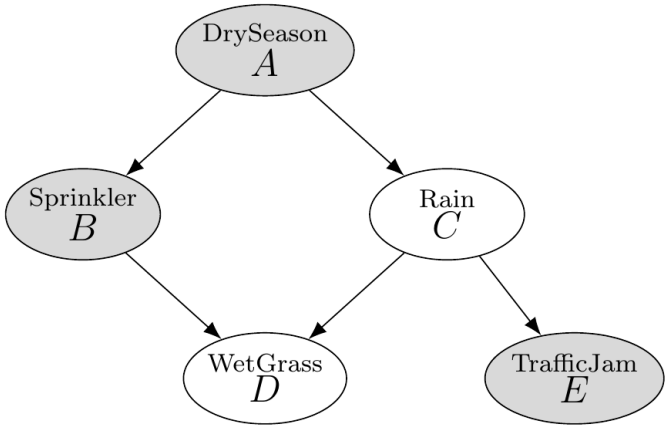
C	$\Pr(C, D = \mathtt{t})$
t	0.3305
f	0.3565

$\xrightarrow{\text{normalise}}$

C	$\Pr(C \mid D = \mathtt{t})$
t	0.48
f	0.52

- The same posterior as before (Rain given WetGrass), now by local operations; normalisation is the only global step.
- **Pruning:** a variable that is neither queried nor observed and has no observed descendant can be dropped before starting: here E ($\Sigma_E \phi^E = 1$).

The elimination order matters



Same query Pr(C, D), two orders.

Eliminating C first multiplies the three factors that mention C and produces a factor over everything else.

order A, B, E : resulting factors $\phi^1(B, C),\ \phi^2(C, D),\ \phi^3(C, D)$ — 2 variables, 2^2 entries each

order C first : $\sum_C \phi^C \odot \phi^D \odot \phi^E = \psi(A, B, D, E)$ — 4 variables, 2^4 entries

- **Width of an order** = size of the largest factor it produces; VE costs O(n · exp width). The best width over all orders is the treewidth of the moralised graph; finding it is NP-hard, greedy heuristics (min-fill, min-degree) work well in practice.

Other inference algorithms

- Exact inference costs $O(n \cdot \exp w)$, w the width. **It is NP-hard in general** [Cooper 90]; when the width is large, it is out of reach.
  - To answer multiple queries, we use the **message passing** algorithm.
- **Approximate inference** is an alternative.
  - **Sampling.** Draw samples from the network and estimate the query by frequencies: forward sampling, likelihood weighting, MCMC (Gibbs, Metropolis–Hastings).
  - **Variational.** Replace the posterior by the closest member of a tractable family: mean field, loopy belief propagation (message passing on a graph that is not a tree).
  - Approximation with guarantees is NP-hard too [Dagum, Luby 93]; in practice the methods work well, and are the default in probabilistic programming languages.
- **This course: exact inference only**.

Where to read

- J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, 1988; *Causality*, 2009.
- A. Darwiche, *Modeling and Reasoning with Bayesian Networks*, CUP 2009 — the running example and the algorithms of this lecture (Ch. 6 variable elimination, Ch. 7 jointrees).
- D. Koller, N. Friedman, *Probabilistic Graphical Models*, MIT Press 2009 — Ch. 6 template models, Ch. 9–10 exact inference.
- J.-W. van de Meent, B. Paige, H. Yang, F. Wood, *An Introduction to Probabilistic Programming*, 2018 — Ch. 3, compiling programs to graphical models.
- B. Jacobs, F. Zanasi, *A predicate/state transformer semantics for Bayesian learning*, MFPS 2016; *The logical essentials of Bayesian reasoning*, 2020 — channels, conditioning and inference in categorical form.

Beyond first order

probabilistic programs as stochastic models

A BN is a first-order program

$$
\begin{array}{l}
\textbf{let } d = \texttt{bernoulli}_{0.6} \textbf{ in} \\
\textbf{let } s = \textbf{case } d \textbf{ of } \{\texttt{t} \Rightarrow \texttt{bernoulli}_{0.8};\ \texttt{f} \Rightarrow \texttt{bernoulli}_{0.1}\} \textbf{ in} \\
\textbf{let } r = \textbf{case } d \textbf{ of } \{\texttt{t} \Rightarrow \texttt{bernoulli}_{0.2};\ \texttt{f} \Rightarrow \texttt{bernoulli}_{0.75}\} \textbf{ in} \\
\textbf{let } w = \textbf{case } \langle s, r\rangle \textbf{ of } \{\texttt{tt} \Rightarrow \texttt{bernoulli}_{0.99};\ \texttt{tf} \Rightarrow \texttt{bernoulli}_{0.9}; \\
\qquad\qquad\qquad\qquad\qquad \texttt{ft} \Rightarrow \texttt{bernoulli}_{0.7};\ \ \texttt{ff} \Rightarrow \texttt{bernoulli}_{0.01}\} \textbf{ in} \\
\langle d, s, r, w\rangle
\end{array}
$$

bernoulli samples from a distribution; **case** encodes a CPT.

The term defines the joint over ⟨d, s, r, w⟩.

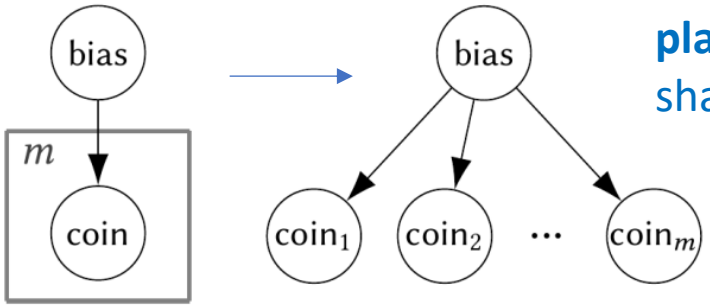
- **Modelling with BNs is like programming with circuits:** no abstraction, no repetition, no recursion. Every variable is written out by hand.

- **Probabilistic programs** are **stochastic models.** PPLs usually separate the description of the model (the program) from the computation of its distribution (inference).

In [POPL24]: **probabilistic programs *are* higher order BNs**, bringing expressivity and compositionality

Plates and dynamic Bayesian networks

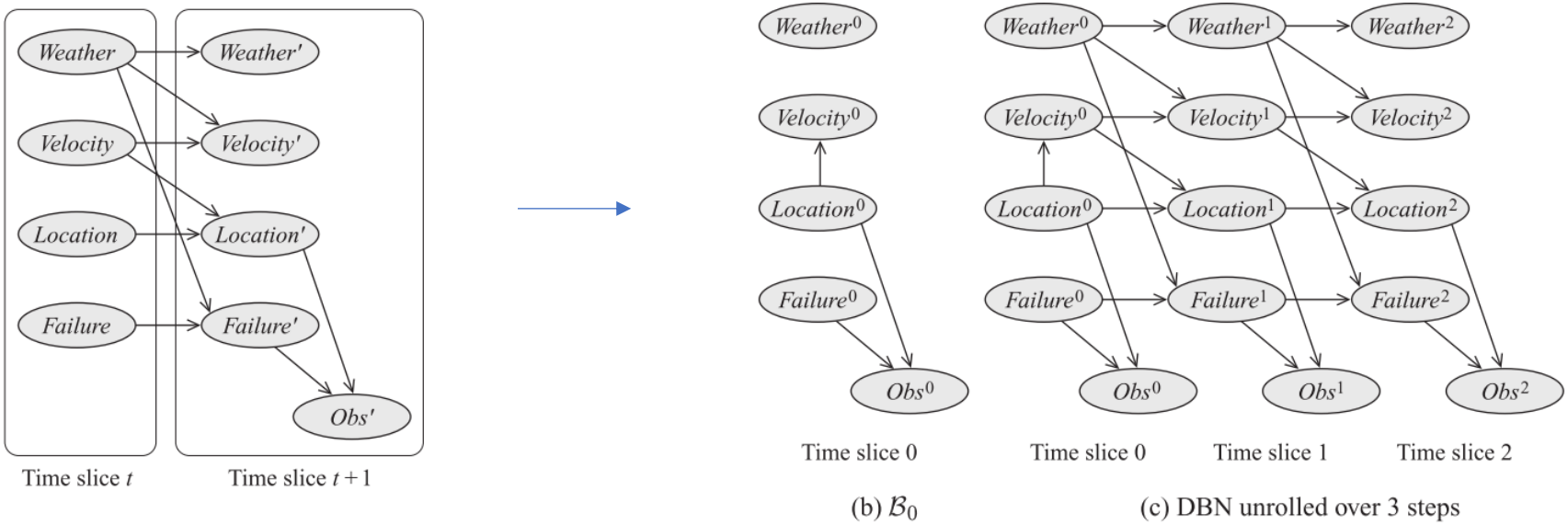
Tossing a coin:



Meta-formalism
plate: repeated structure and shared parameters

Monitoring a vehicle:

Dynamic Bayesian networks are an instance extensively used in real-world applications—*e.g.* mobile robotics.



Repetition and sharing

Two tosses of one coin of unknown bias: the bias is sampled once and shared; the toss is repeated. A naive term gets one of the two wrong.

call-by-value: shares the toss

$$\textbf{let}\ bias = \texttt{sample}_d\ \textbf{in let}\ coin = \textbf{case}\ bias\ \textbf{of}\{\cdots\}\ \textbf{in}\ \langle coin, coin\rangle$$

call-by-name: copies the bias too

$$\textbf{let}\ bias = \texttt{sample}_d\ \textbf{in}\ \langle \textbf{case}\ bias\ \textbf{of}\{\cdots\},\ \textbf{case}\ bias\ \textbf{of}\{\cdots\}\rangle$$

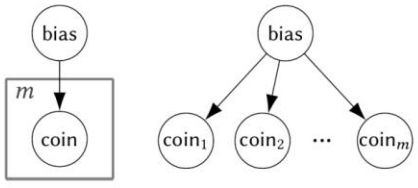
share the bias, copy the toss

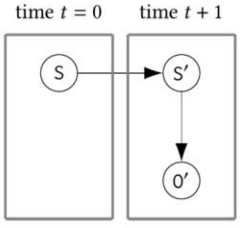
$$\textbf{let}\ bias = \texttt{sample}_d\ \textbf{in let}\ coin = !(\textbf{case}\ bias\ \textbf{of}\{\cdots\})\ \textbf{in}\ \langle \textbf{der}\, coin,\ \textbf{der}\, coin\rangle$$

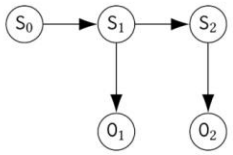
$$\textbf{let}\ bias = \texttt{sample}_d\ \textbf{in let}\ coin = !(\textbf{case}\ bias\ \textbf{of}\{\cdots\})\ \textbf{in}\ \langle \textbf{der}\, coin, \textbf{der}\, coin\rangle$$
$$\to^* \quad \textbf{let}\ bias = \texttt{sample}_d\ \textbf{in let}\ y_1 = \textbf{case}\ bias\ \textbf{of}\{\cdots\}\ \textbf{in let}\ y_2 = \textbf{case}\ bias\ \textbf{of}\{\cdots\}\ \textbf{in}\ \langle y_1, y_2\rangle$$

- **Call-by-push-value:** ! (thunk) delays a computation so that it can be copied, der (force) runs it. Reduction unrolls the term into the first-order program of the two-toss network — without sampling.

Every program of ground type is a BN







Theorem (POPL 24). Every closed term of ground type reduces to a first-order normal form, i.e. a Bayesian network; conversely every BN is a term.

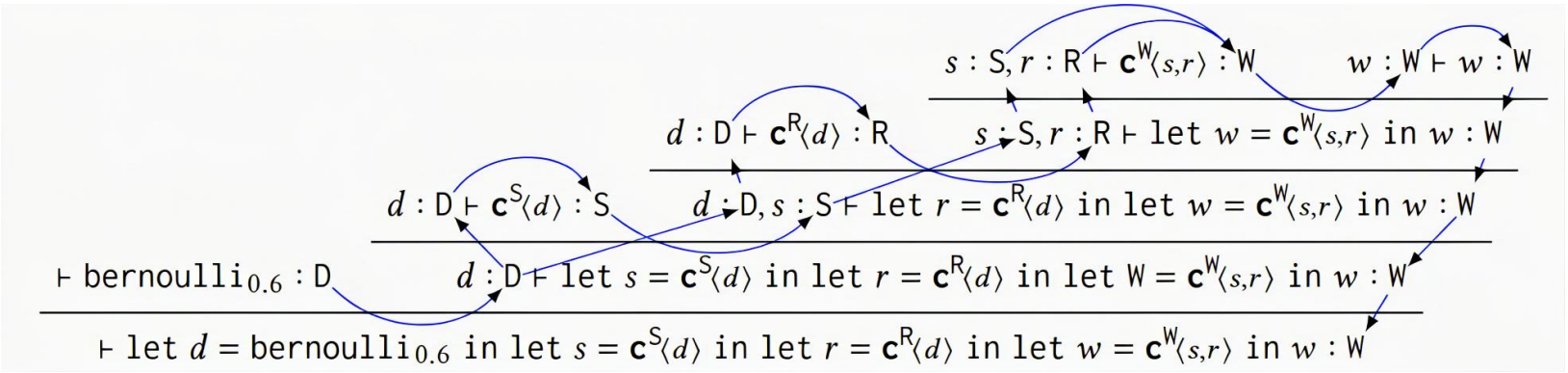
Recursion (fix, right) unrolls a dynamic BN: the type depends on n.

$$
\begin{array}{ll}
\texttt{let} & \mathsf{bias} = \mathtt{sample}_d \ \texttt{in} \\
\texttt{let} & \mathsf{coin} = \,!\,(\mathsf{c}\langle \mathsf{bias} \rangle) \ \texttt{in} \\
\texttt{let} & y_1 = \mathtt{der}\,\mathsf{coin} \ \texttt{in} \\
\texttt{let} & y_2 = \mathtt{der}\,\mathsf{coin} \\
\texttt{in} & \langle y_1, y_2 \rangle
\end{array}
$$

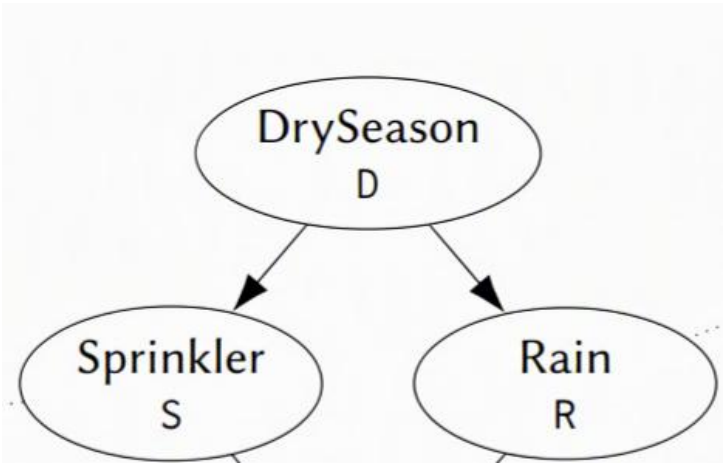
$$
\begin{array}{lll}
t & \triangleq & \lambda n.\texttt{let}\ s_0 = \mathtt{bern}_p\ \texttt{in}\ u\ n\ s_0 \\
u & \triangleq & \texttt{fix}\ !(\lambda x.\lambda n.\lambda s.\texttt{if}\ \mathtt{isZero}(n) \\
 & & \quad \texttt{then}\ \ s \\
 & & \quad \texttt{else}\ \ \texttt{let}\ s' = \mathsf{c}^{\mathsf{S}}\langle s \rangle\ \texttt{in} \\
 & & \qquad\ \ \texttt{let}\ o' = \mathsf{c}^{\mathsf{O}}\langle s' \rangle\ \texttt{in} \\
 & & \qquad\ \ \texttt{let}\ m = \mathtt{pred}(n)\ \texttt{in} \\
 & & \qquad\ \ \texttt{let}\ r = (\mathtt{der}\,x)\ m\ s' \\
 & & \qquad\ \ \texttt{in}\ \ \langle o', r \rangle\ )
\end{array}
$$

$\vdash\ t\ :\ X_1 \otimes \cdots \otimes X_k\ \diamond\ \phi$ the type says which random variables the term defines; ϕ is the distribution over them

Type derivations as BN



- The (intersection) type system tracks the random variables a term generates: the type says which network, the semantics ϕ is its distribution, both computed compositionally from the derivation.



Break

10 minutes

Quantum Bayesian Networks

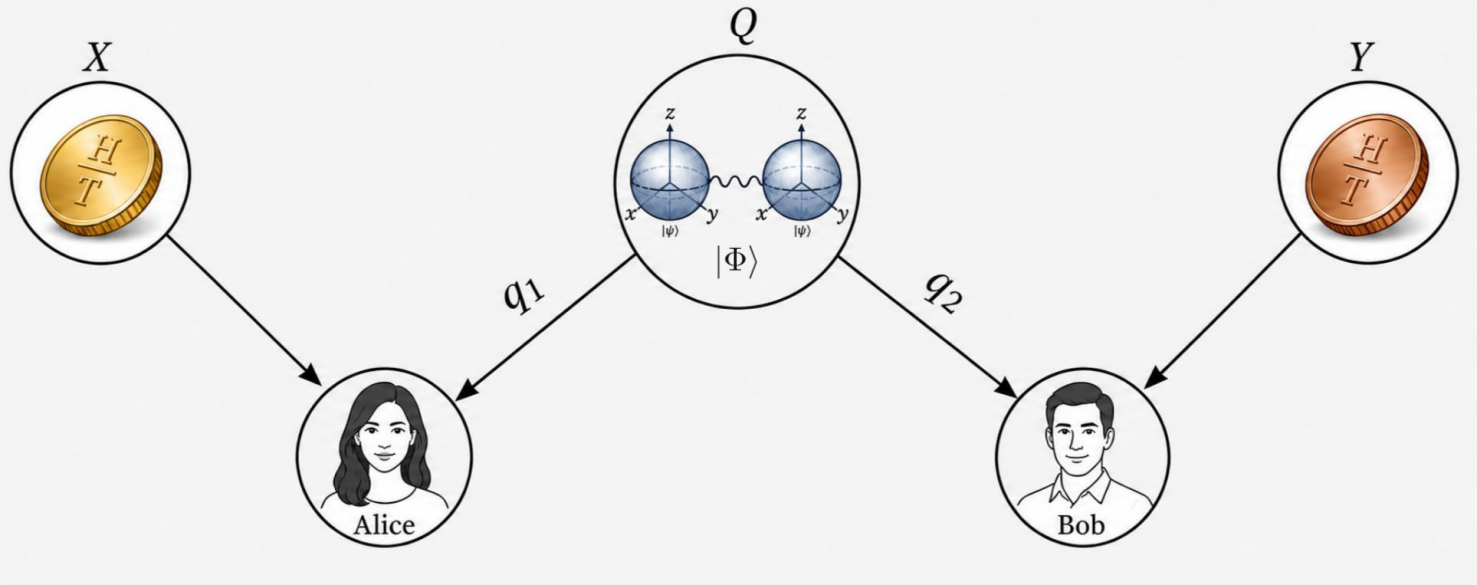
Theory-independent limits on correlations from generalised Bayesian networks

Joe Henson Raymond Lal Matthew F. Pusey

A mathematical framework to describe causal relations, to analyse correlations, and to predict the probabilities of measurement outcomes, in **systems involving both classical and quantum data.**

Bell set-up

Back to the beginning of the class



Q = *Preparation of the entangled* q_1 *and* q_2

X = *Flip a coin*

Alice = *Measure on* q_1 *parameterized by the value of* X

Alice measures q_1 in the computational basis when $X = \mathrm{h}$, in the Hadamard one otherwise. Bob does the same with q_2 and Y.

A **QBN** is a DAG with **Quantum Factors** associated to it

States, channels, measurements

state $\rho \in \mathcal{D}(\mathcal{H}) : \ \rho \geq 0, \ \mathrm{tr}(\rho) = 1$
$|\psi\rangle\langle\psi|$ pure, $\sum_i p_i \rho_i$ mixed

$\rho \in \mathcal{L}(\mathcal{H})^+$

channel $\mathcal{E} : \mathcal{L}(\mathcal{H}_1) \to \mathcal{L}(\mathcal{H}_2)$ linear, completely positive, trace preserving
$\rho \mapsto U\rho U^\dagger$ (unitary), $\mathbb{C} \to \mathcal{L}(\mathcal{H})$ (preparation), $\mathrm{tr}_{\mathcal{H}}$ (discard)

measurement $\{E_x\}_{x \in O}$ with $E_x \geq 0$ and $\sum_x E_x = \mathrm{Id}_{\mathcal{H}}$ (POVM)
$\Pr(x) = \mathrm{tr}(E_x \rho)$ (Born rule)

$E_x \in \mathcal{L}(\mathcal{H})^+$

zero qubits $\mathcal{H}^{\otimes 0} = \mathbb{C}, \quad \mathcal{L}(\mathbb{C})^+ = \mathbb{R}_0^+$
a POVM on the empty register is a probability distribution $(E_x)_x$: sampling

- **Running example:** the Bell state $|\Phi\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$, and the two measurements std = $\{|0\rangle\langle 0|, |1\rangle\langle 1|\}$ and had = $\{|+\rangle\langle +|, |-\rangle\langle -|\}$.

The **Choi–Jamiołkowski isomorphism:**
From $\mathcal{E}: \mathcal{L}(\mathcal{H}_1)^+ \to \mathcal{L}(\mathcal{H}_2)^+$ to $\mathcal{J}(\mathcal{E}) \in \mathcal{L}(\mathcal{H}_1 \otimes \mathcal{H}_2)^+$

$\mathcal{J}(\mathcal{E}) \triangleq \sum_{i,j} |i\rangle\langle j| \otimes \mathcal{E}(|i\rangle\langle j|)$ $\qquad$ $\mathcal{E}$ is a channel $\iff$ $\mathcal{J}(\mathcal{E}) \geq 0$ and $\mathrm{tr}_{\mathcal{H}_2} \mathcal{J}(\mathcal{E}) = \mathrm{Id}_{\mathcal{H}_1}$

Quantum factors

Classical factor over r.v.s $\mathbf{X}$	Quantum factor over r.v.s $\mathbf{X}$ and the quantum register Q
$\phi : \mathsf{Val}(\mathbf{X}) \to \mathbb{R}_0^+$	$\phi : \mathsf{Val}(\mathbf{X}) \to \mathcal{L}(\mathcal{H}_Q)^+$

$\phi : \mathsf{Val}(\mathbf{X}) \to \mathcal{L}(\mathcal{H}_Q)^+$ a positive operator on $\mathcal{H}_Q = \bigotimes_{q \in Q} \mathcal{H}_q$ for each tuple of classical values

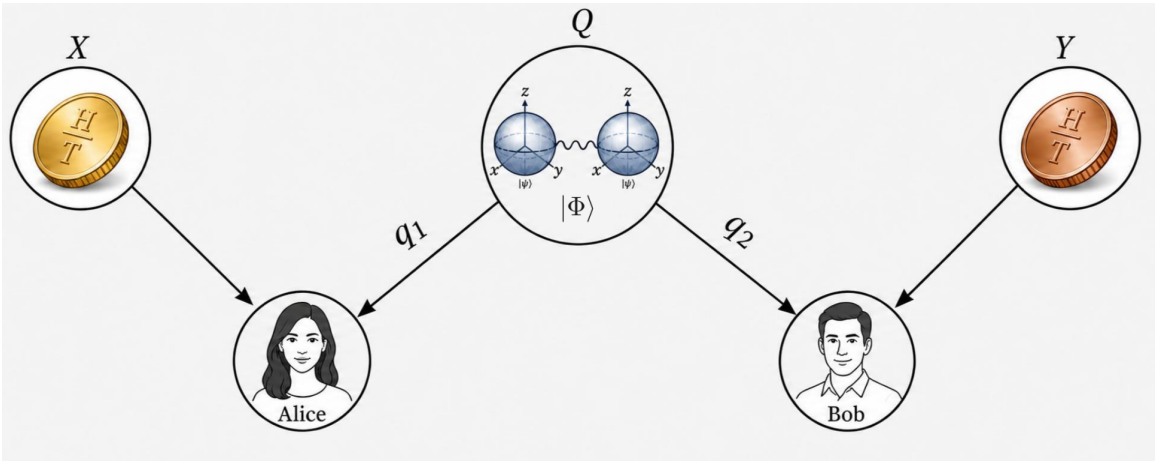
$Q = \emptyset$: $\mathcal{L}(\mathbb{C})^+ = \mathbb{R}_0^+$, a classical factor
$\mathbf{X} = \emptyset$: a positive operator on $\mathcal{H}_Q$, e.g. a state $\rho \in \mathcal{D}(\mathcal{H}_Q)$

- **Conservative extension:** classical factors are the Q-factors with no qubits; states and positive operators are the Q-factors with no classical names.
- **Challenge:** reconcile sharing of classical values with no-cloning of quantum data in one product.

The Bell set-up

	$\mathcal{H}_{q_1,q_2}$
$*$	$\lvert\Phi^+\rangle\langle\Phi^+\rvert$

X	$\mathbb{C}$
t	0.5
f	0.5

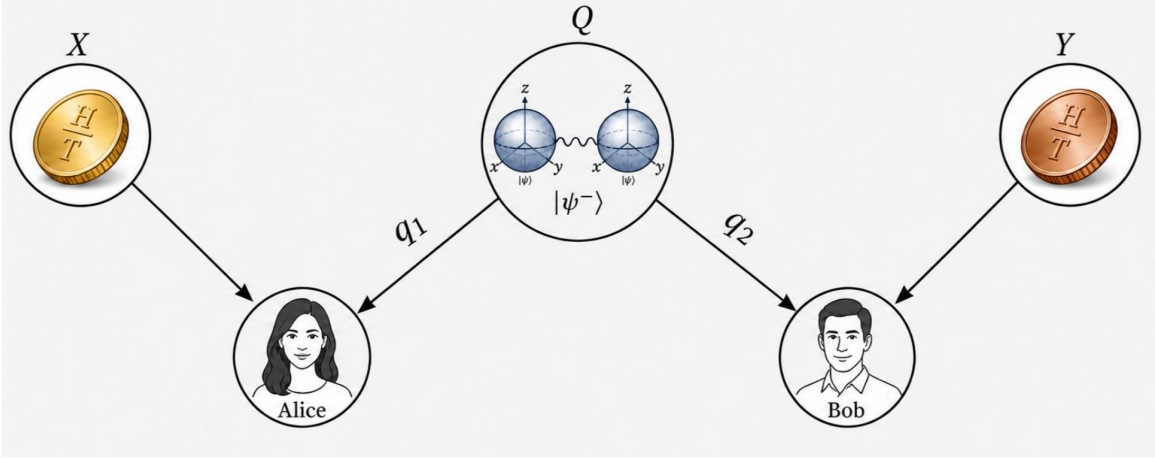


Y	$\mathbb{C}$
t	0.5
f	0.5

X	A	$\mathcal{H}_{q_1}$
t	t	$\lvert 0\rangle\langle 0\rvert$
t	f	$\lvert 1\rangle\langle 1\rvert$
f	t	$\lvert +\rangle\langle +\rvert$
f	f	$\lvert -\rangle\langle -\rvert$

Y	B	$\mathcal{H}_{q_2}$
t	t	$\lvert 0\rangle\langle 0\rvert$
t	f	$\lvert 1\rangle\langle 1\rvert$
f	t	$\lvert +\rangle\langle +\rvert$
f	f	$\lvert -\rangle\langle -\rvert$

Zoom in: Alice's node



Given x = t Alice measures q_1 in the computational basis, given x = f in the Hadamard basis. Her node is labelled by the Q-factor ϕ^A for A given ({X}, {q_1}): for each x a measurement of q_1, one effect per outcome a:

X	A	$\mathcal{H}_{q_1}$
t	t	$\|0\rangle\langle 0\|$
t	f	$\|1\rangle\langle 1\|$
f	t	$\|+\rangle\langle +\|$
f	f	$\|-\rangle\langle -\|$

$$\phi^A(\mathtt{t},\mathtt{t}) = |0\rangle\langle 0| = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \qquad \phi^A(\mathtt{t},\mathtt{f}) = |1\rangle\langle 1| = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\phi^A(\mathtt{f},\mathtt{t}) = |+\rangle\langle +| = \tfrac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \qquad \phi^A(\mathtt{f},\mathtt{f}) = |-\rangle\langle -| = \tfrac{1}{2}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

Sum-out and Product

Two kind of sum-outs:

$$\text{classical name } Z: \quad \left(\sum_Z \phi\right)(\mathbf{y}) = \sum_z \phi(\mathbf{y}z) \qquad (\phi \text{ over } \mathbf{Y} \uplus \{Z\})$$

$$\text{register } Q' \subseteq Q: \quad \left(\sum_{Q'} \phi\right)(\mathbf{x}) = \mathrm{tr}_{\mathcal{H}_{Q'}}\left(\phi(\mathbf{x})\right)$$

The product over shared classical and quantum data:

$$\phi_1 \text{ over } (\mathbf{X} \uplus \mathbf{Z},\ Q \uplus S), \quad \phi_2 \text{ over } (\mathbf{Z} \uplus \mathbf{Y},\ S \uplus R): \qquad \phi_1 \odot \phi_2 \text{ over } (\mathbf{X} \uplus \mathbf{Z} \uplus \mathbf{Y},\ Q \uplus R)$$

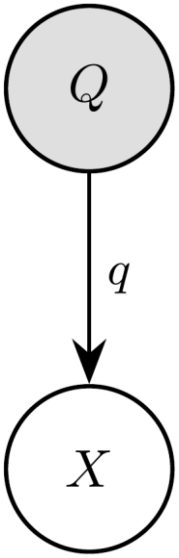
$$(\phi_1 \odot \phi_2)(\mathbf{xyz}) \triangleq \langle \mathrm{M} | \left(\phi_1(\mathbf{xy}) \otimes \phi_2(\mathbf{yz})\right) | \mathrm{M} \rangle$$

Where $|\mathrm{M}\rangle = \sum_{i=1}^{n} |ii\rangle$ is the maximaly-entangled vector on $\mathcal{H}_Q \otimes \mathcal{H}_Q$

- **Shared classical name:** takes one value in both factors and stays in the scope (sharing, as for factors). Classical scope of the product: the union.
- **Shared qubit:** the qubit disappears (contraction, as for tensor networks). Quantum scope of the product: the symmetric difference.

Examples

Measurement



$\Pr(\mathrm{x}) = tr(E_{\mathrm{x}}\rho)$

	$\mathcal{H}_q$
$*$	ρ

$\odot$

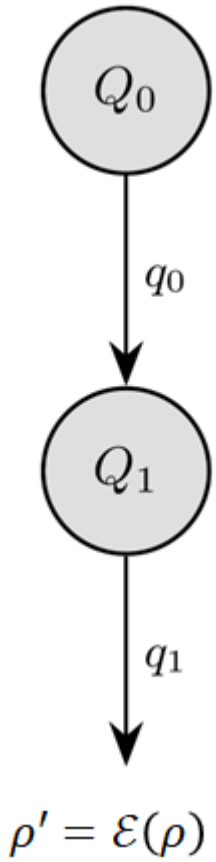
x	$\mathcal{H}_q$
0	E_0^t
1	E_1^t

$=$

x	$\mathbb{C}$
0	$\mathrm{tr}(E_0\rho)$
1	$\mathrm{tr}(E_1\rho)$

Examples

Quantum Channel



$\rho' = \mathcal{E}(\rho)$

	$\mathcal{H}_{q_1}$
$*$	ρ

$\odot$

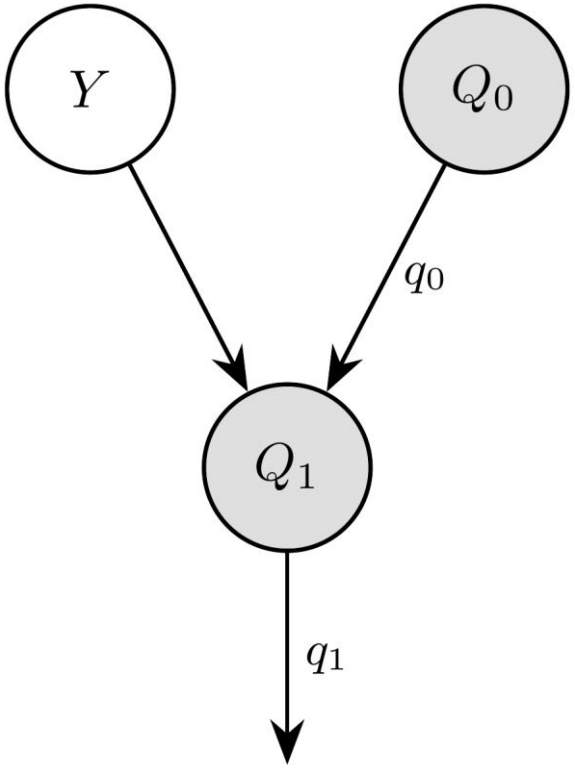
	$\mathcal{H}_{q_1,q_2}$
$*$	$\mathcal{J}(\mathcal{E})$

$=$

	$\mathcal{H}_{q_2}$
$*$	$\mathcal{E}(\rho)$

Examples

Controlled Channel



$\rho' = \mathcal{E}_y(\rho)$

	$\mathcal{H}_q$
$*$	ρ

$\odot$

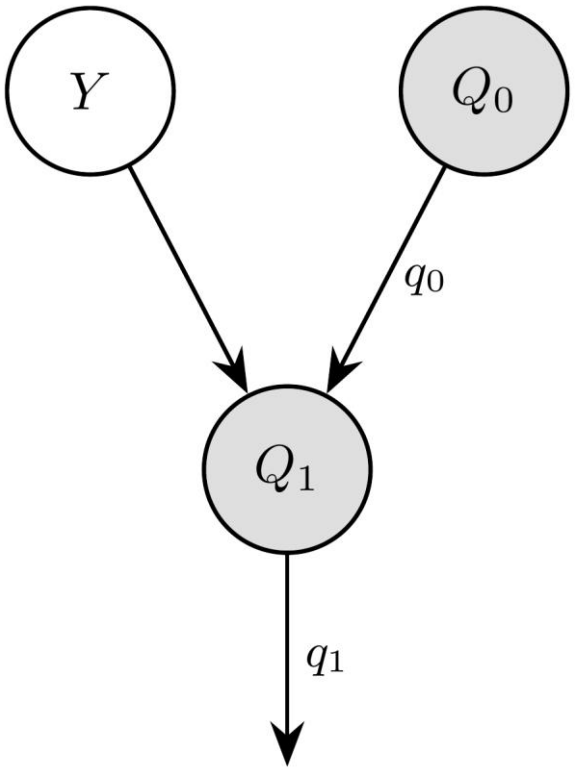
Y	$\mathcal{H}_{q_1,q_2}$
0	$\mathcal{J}(\mathcal{E}_0)$
1	$\mathcal{J}(\mathcal{E}_1)$

$=$

Y	$\mathcal{H}_{q_2}$
0	$\mathcal{E}_0(\rho)$
1	$\mathcal{E}_1(\rho)$

Examples

Controlled Channel



$\rho' = \mathcal{E}_y(\rho)$

Y	$\mathcal{H}_{q_2}$
0	$\mathcal{E}_0(\rho)$
1	$\mathcal{E}_1(\rho)$

$\odot$

Y	$\mathbb{C}$
0	0.7
1	0.3

$=$

Y	$\mathcal{H}_{q_2}$
0	$0.7 \cdot \mathcal{E}_0(\rho)$
1	$0.3 \cdot \mathcal{E}_1(\rho)$

Entries

$$\phi(\mathbf{x})[\mathbf{i},\mathbf{j}] \triangleq \langle \mathbf{i}|\, \phi(\mathbf{x})\, |\mathbf{j}\rangle \qquad \mathbf{i},\mathbf{j} \in \{0,1\}^Q$$

Sum-out, via entries:

$$\text{classical name } Z: \qquad \big(\textstyle\sum_Z \phi\big)(\mathbf{y})[\mathbf{i},\mathbf{j}] = \sum_z \phi(\mathbf{y}z)[\mathbf{i},\mathbf{j}]$$

$$\text{register } Q' \subseteq Q: \qquad \big(\textstyle\sum_{Q'} \phi\big)(\mathbf{x})[\mathbf{i},\mathbf{j}] = \sum_{\mathbf{k}\in\{0,1\}^{Q'}} \phi(\mathbf{x})[\mathbf{ik},\mathbf{jk}] \;=\; \mathrm{tr}_{\mathcal{H}_{Q'}}\big(\phi(\mathbf{x})\big)[\mathbf{i},\mathbf{j}]$$

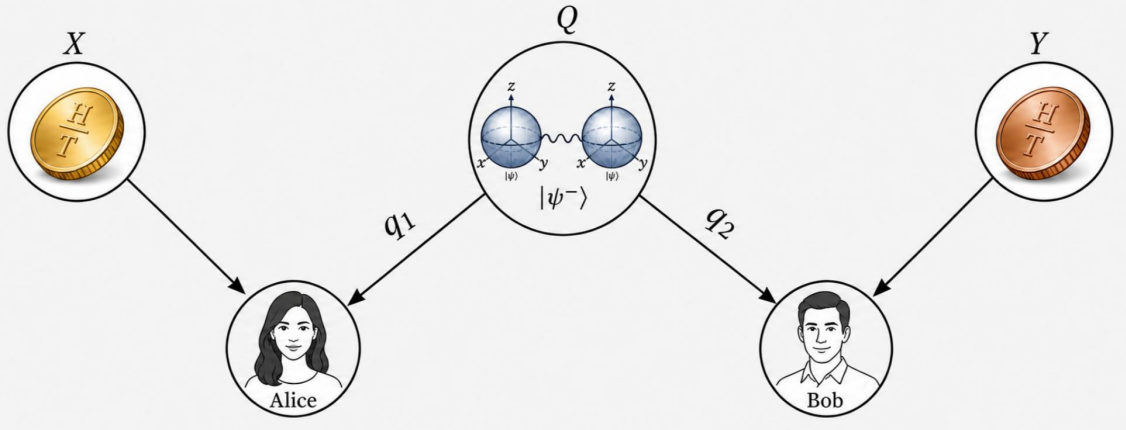
Factor product does not require the tensor product if we focus on entries.

$$\phi_1 \text{ over } (\mathbf{X}\uplus\mathbf{Z},\ Q\uplus S), \quad \phi_2 \text{ over } (\mathbf{Z}\uplus\mathbf{Y},\ S\uplus R): \qquad \phi_1\odot\phi_2 \text{ over } (\mathbf{X}\uplus\mathbf{Z}\uplus\mathbf{Y},\ Q\uplus R)$$

$$(\phi_1\odot\phi_2)(\mathbf{xzy})[\mathbf{i}_Q\mathbf{i}_R,\mathbf{j}_Q\mathbf{j}_R] \triangleq \sum_{\mathbf{i}_S,\mathbf{j}_S\in\{0,1\}^S} \phi_1(\mathbf{xz})[\mathbf{i}_Q\mathbf{i}_S,\mathbf{j}_Q\mathbf{j}_S]\cdot\phi_2(\mathbf{zy})[\mathbf{i}_S\mathbf{i}_R,\mathbf{j}_S\mathbf{j}_R]$$

A Q-factor is an array with one axis per classical name and two per qubit: the data structure of numerical libraries.

QBNs (via quantum factors)



A family of instruments indexed by classical values is a Q-cpt, and conversely.

Definition (Quantum Bayesian network, via quantum factors) [FSCD 26]

A quantum Bayesian network over a finite set of random variables $\mathbf{X}$ and a finite set of disjoint quantum registers $\mathbf{Q}$ is a pair $(\mathcal{G}, \Phi)$ where

- $\mathcal{G}$ is a DAG whose set of nodes is $\mathbf{X} \cup \mathbf{Q}$; each out-edge of a classical node X is labelled by X, and the out-edges of a quantum node Q are labelled by non-empty registers partitioning Q;
- Φ assigns to each node V a **quantum CPT** (Q-cpt) for V given the random variables and qubits labelling its in-edges.

Semantics

$[\![\mathcal{B}]\!] \triangleq \bigodot_{V \in \mathbf{X} \cup \mathbf{Q}} \Phi(V)$ is a probability distribution over $\mathbf{X}$.

Quantum CPTs

$$\text{Q-cpt for } Z \text{ given } (\mathbf{Y}, Q): \quad \phi \text{ over } (\{Z\} \uplus \mathbf{Y}, Q), \quad \big(\textstyle\sum_Z \phi\big)(\mathbf{y}) = \mathrm{Id}_{\mathcal{H}_Q} \quad \forall \mathbf{y}$$

$$\text{Q-cpt for } Q' \text{ given } (\mathbf{X}, Q''): \quad \phi \text{ over } (\mathbf{X}, Q' \uplus Q''), \quad \big(\textstyle\sum_{Q'} \phi\big)(\mathbf{x}) = \mathrm{Id}_{\mathcal{H}_{Q''}} \quad \forall \mathbf{x}$$

$$\text{Q-cpts for } Q' \text{ given } (\mathbf{Y}, Q'') \longleftrightarrow \text{families of channels } (\mathcal{E}_{\mathbf{y}})_{\mathbf{y}} \text{ from } \mathcal{H}_{Q''} \text{ to } \mathcal{H}_{Q'}: \quad \phi(\mathbf{y}) = \mathcal{J}(\mathcal{E}_{\mathbf{y}})$$

$$\text{Q-cpts for } Z \text{ given } (\mathbf{Y}, Q) \longleftrightarrow \text{families of measurements } (\{E_{z,\mathbf{y}}\}_z)_{\mathbf{y}} \text{ on } \mathcal{H}_Q: \quad \phi(z, \mathbf{y}) = E_{z,\mathbf{y}}^T$$

consequence: $\phi \odot \rho = \mathcal{E}(\rho)$ $\qquad$ $\phi' \odot \rho = \big(z \mapsto \mathrm{tr}(E_z\, \rho)\big)$

The product applies the channel and implements the Born rule: Q-cpts are the Choi images of channels and of measurements, indexed by the values of the classical parents

Limit cases: a CPT (no qubits); a Choi state (no classical names).

Dealing with hybrid data, preserving classical Bayesian theory

- No qubits: the product and sum-out of classical factors.
- No classical names: tensor contraction and partial trace.

These properties is all what Variable Elimination and Message Passing need!

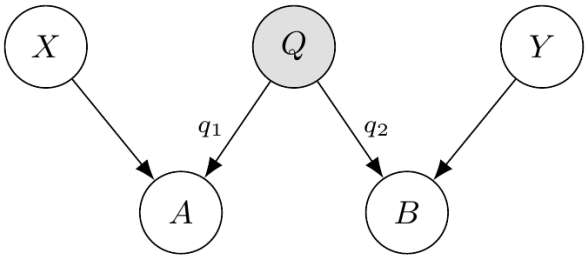
Sum & Product do satisfy:

- **Associativity** (the condition is always satisfied by QBNs)
- **Distributivity** (under usual conditions)

$$\phi_1 \odot \phi_2 = \phi_2 \odot \phi_1 \qquad (\phi_1 \odot \phi_2) \odot \phi_3 = \phi_1 \odot (\phi_2 \odot \phi_3) \text{ if } Q_1 \cap Q_2 \cap Q_3 = \emptyset$$

$$\sum_U \sum_V \phi = \sum_V \sum_U \phi \qquad \sum_V (\phi_1 \odot \phi_2) = \left(\sum_V \phi_1\right) \odot \phi_2 \text{ if } V \notin \mathrm{Nm}(\phi_2)$$

The Bell set-up, computed



Five Q-cpts: X, Y ↦ 1/2; Q ↦ $|\Phi\rangle\langle\Phi|$; A ↦ $\phi^A(a, x)$ of the previous slides; B likewise with Y and q_2.

The product contracts q_1 and q_2; nothing is left to sum out.

$$\Pr(x, y, a, b) \;=\; \phi^X \odot \phi^Y \odot \phi^Q \odot \phi^A \odot \phi^B \;=\; \tfrac{1}{4}\,\mathrm{tr}\Big(\big(\phi^A(a, x) \otimes \phi^B(b, y)\big)\,|\Phi\rangle\langle\Phi|\Big) \qquad |\Phi\rangle = \tfrac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

$\Pr(a, b \mid x, y)$:

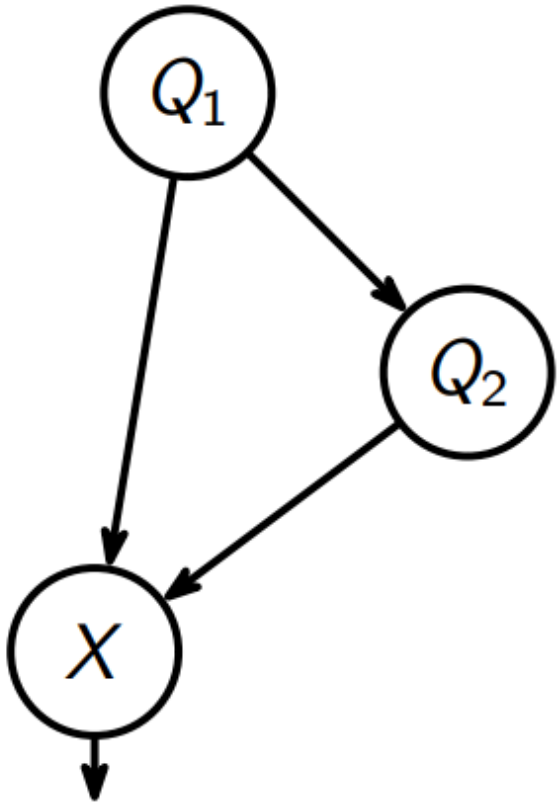
	$x = y$		$x \neq y$	
$a \backslash b$	t	f	t	f
t	$\frac{1}{2}$	0	$\frac{1}{4}$	$\frac{1}{4}$
f	0	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$

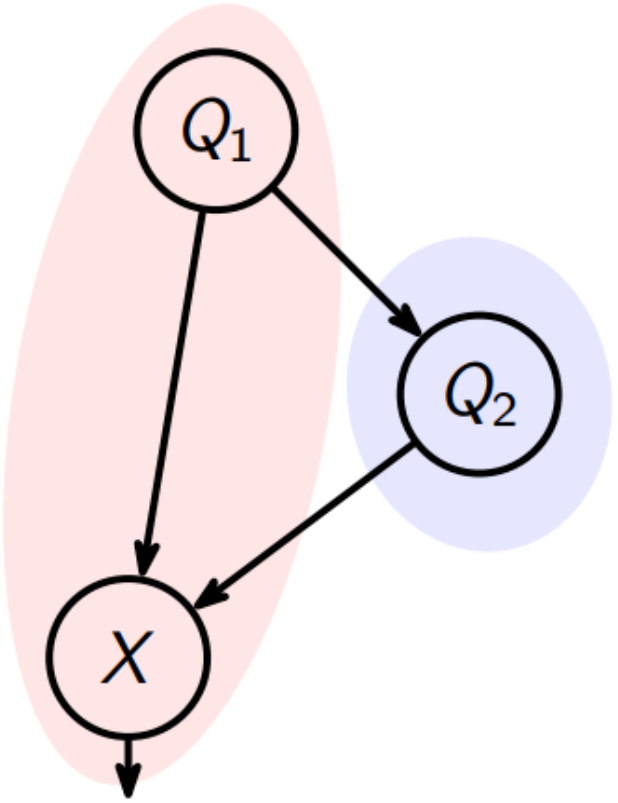
same basis: perfectly correlated outcomes; different bases: uniform.

$$[\![\mathcal{B}]\!] \;=\; \sum_{\mathrm{Nm}(\mathcal{B}_1) \cup \mathrm{Nm}(\mathcal{B}_2) \,-\, \mathrm{Nm}(\mathcal{B})} \big([\![\mathcal{B}_1]\!] \odot [\![\mathcal{B}_2]\!]\big)$$

Compositionality: a sub-network is interpreted the same way, and the whole is the product of the parts.

Compositionality of the semantics



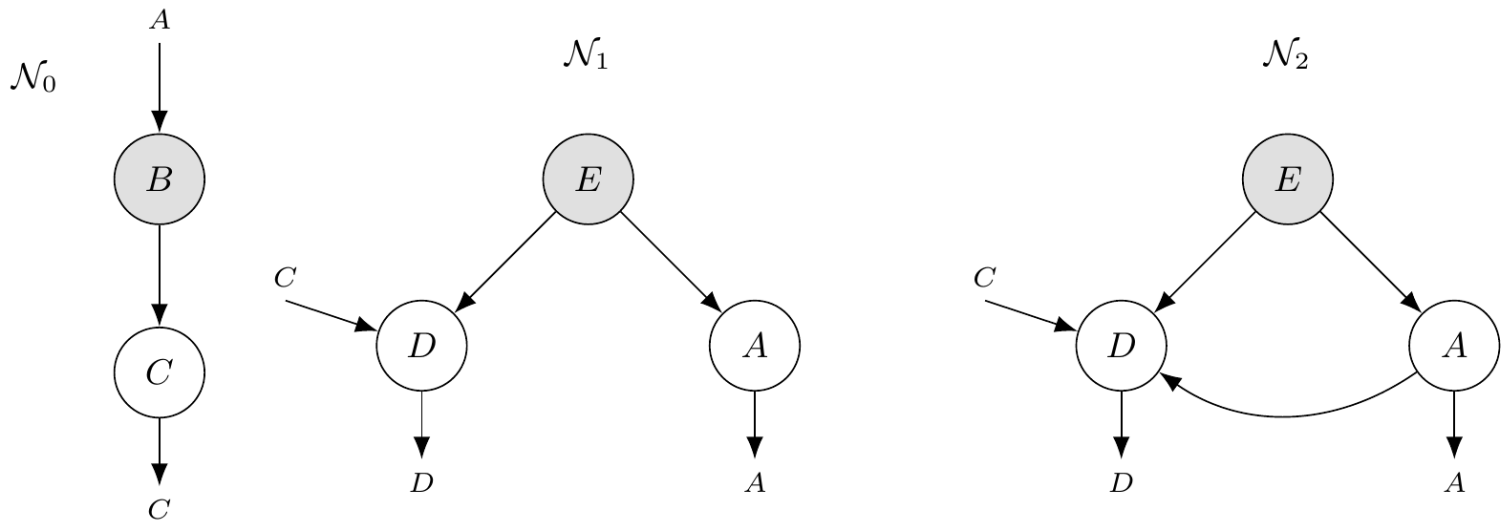


We move to quantum factors bcs HLP semantics is global: instruments are composed in a fixed top-down order; no interpretation of a sub-network, no sum-out of hidden variables by components.

Perks of typing

Do parts compose?

N_0 awaits an input A and outputs C; N_1 and N_2 await C and output A and D. N_0 plugged into N_1 is a DAG; N_0 plugged into N_2 has a directed cycle.



$\mathcal{N}_0 \vdash A \multimap C$ $\quad$ $\mathcal{N}_1 \vdash (A \multimap C) \multimap D$ $\quad$ $\mathcal{N}_2 \vdash C \multimap (A \otimes D)$

N_2 can be given no type matching that of N_0!

A typed syntax for QBNs exists (FSCD 26); not shown here.

Pointers: quantum theory as inference

- M. Leifer, R. Spekkens, *Towards a formulation of quantum theory as a causally neutral theory of Bayesian inference*, Phys. Rev. A 88, 2013 — quantum conditional states; channel application, Born rule and composition as belief propagation; a quantum Bayes theorem. The basis of QBNs.
- B. Coecke, R. Spekkens, *Picturing classical and quantum Bayesian inference*, Synthese 186, 2012 — the diagrammatic version.
- B. Jacobs, *Lower and upper conditioning in quantum Bayesian theory*, QPL 2018 / EPTCS 2019 — two forms of quantum conditioning. K. Cho, B. Jacobs, *Disintegration and Bayesian inversion via string diagrams*, MSCS 2019.
- A. Parzygnat, *Inverses, disintegrations, and Bayesian inversion in quantum Markov categories*, 2020; A. Parzygnat, B. Russo, *A non-commutative Bayes' theorem*, LAA 2022; A. Parzygnat, *Conditional distributions for quantum systems*, QPL 2021 — Bayes map vs Petz recovery vs Leifer–Spekkens.

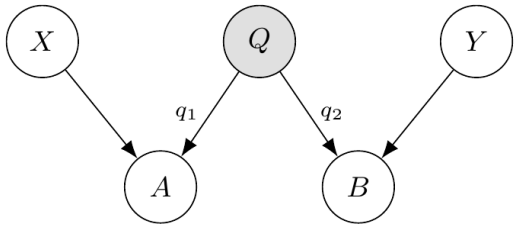
Pointers: QBNs and causal models

- **Quantum Bayesian networks:** R. Tucci 1995 (amplitudes; a different notion); J. Henson, R. Lal, M. Pusey, *Theory-independent limits on correlations from generalised Bayesian networks*, NJP 2014 — the definition used here; Khanna, Maciel Ansanelli, Pusey, Wolfe 2024.
- **Quantum networks (a distinct framework):** G. Chiribella, G. M. D'Ariano, P. Perinotti, *Theoretical framework for quantum networks*, PRA 2009 — quantum combs; they do not encode BNs (HLP §3.1). Higher-order quantum theory: Bisio, Perinotti 2019; Hoffreumon, Oreshkov 2026.
- **Quantum causal models:** T. Fritz 2012; C. Wood, R. Spekkens 2015; F. Costa, S. Shrapnel 2016; J.-M. Allen, J. Barrett, D. Horsman, C. Lee, R. Spekkens, *Quantum common causes and quantum causal models*, PRX 2017; J. Barrett, R. Lorenz, O. Oreshkov, *Quantum causal models*, 2019 — start here.
- **Categorical causal structure (classical and quantum):** A. Kissinger, S. Uijlen, *A categorical semantics for causal structure*, LMCS 2019; W. Simmons, A. Kissinger 2022, 2024.

Some open research

Ongoing: programming QBNs

$$\begin{array}{l}\textbf{let } x = \texttt{bernoulli}_{0.5} \textbf{ in}\\ \textbf{let } y = \texttt{bernoulli}_{0.5} \textbf{ in}\\ \textbf{letp } \langle r_1, r_2\rangle = \texttt{chan}_{\text{bell}}() \textbf{ in}\\ \textbf{let } a = \textbf{c-meas } x \textbf{ of } \{\texttt{t} \Rightarrow \texttt{meas}_{\text{std}}(r_1);\ \texttt{f} \Rightarrow \texttt{meas}_{\text{had}}(r_1)\} \textbf{ in}\\ \textbf{let } b = \textbf{c-meas } y \textbf{ of } \{\texttt{t} \Rightarrow \texttt{meas}_{\text{std}}(r_2);\ \texttt{f} \Rightarrow \texttt{meas}_{\text{had}}(r_2)\} \textbf{ in}\\ \langle x, y, a, b\rangle\end{array}$$



work in progress

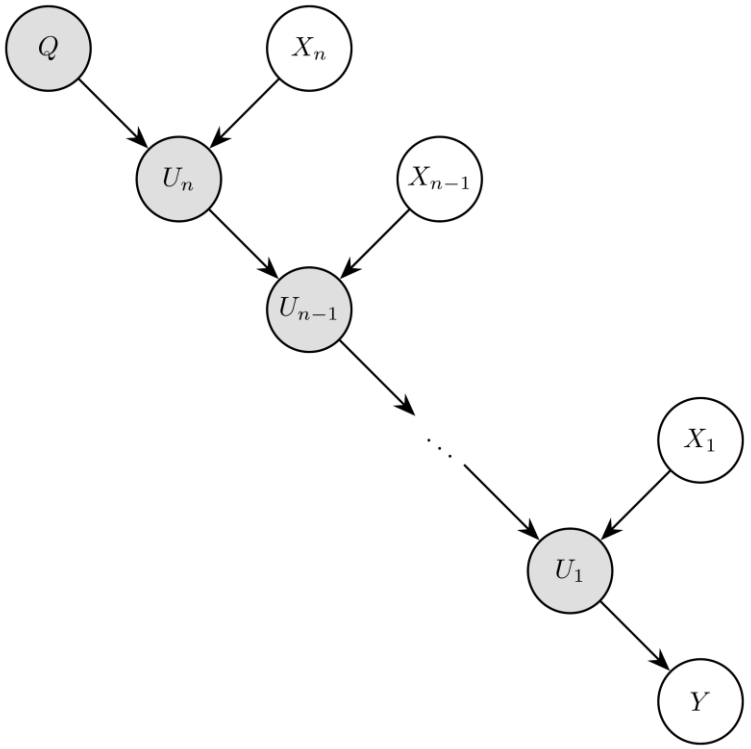
- **Primitives:** measurements and channels, each with a classically controlled version (c-meas, c-chan): a controlled operation is a Q-cpt.
- **A measurement of zero qubits is sample:** the probabilistic language seen earlier is the special case with no qubits.
- **A QBN is a let-term** with no !, der, λ. The names of the typing — X, Y, A, B, q_1, q_2 — are the nodes of the DAG.

$\vdash t_{\text{Bell}} : X \otimes Y \otimes A \otimes B$ axioms: $\vdash \texttt{bernoulli}_{0.5} : X$, $\vdash \texttt{chan}_{\text{bell}}() : q_1 \otimes q_2$, $x{:}X,\ r_1{:}q_1 \vdash \texttt{cm}_{\langle x\rangle}\{r_1\} : A$, …

Ongoing: higher order QBNs

$$t \triangleq \lambda n.\mathsf{let}\ q = \mathsf{chan}_{prep_0}()\ \mathsf{in}\ u\ n\ q$$

$$u \triangleq \mathtt{fix}\ !\big(\lambda f.\lambda n.\lambda q.$$
$$\mathtt{if}\quad \mathsf{isZero}(n)$$
$$\mathtt{then}\quad \mathsf{meas}_{acc}(q)$$
$$\mathtt{else}\quad \mathsf{let}\ x = \mathsf{sample}_d\ \mathsf{in}$$
$$\mathsf{let}\ q' = \mathbf{cc}_{\langle x\rangle\{q\}}\ \mathsf{in}$$
$$\mathsf{let}\ m = \mathsf{pred}(n)\ \mathsf{in}$$
$$(\mathsf{der}\ f)\ m\ q'\ \big)$$



work in progress

Ongoing: semantics, and what is next

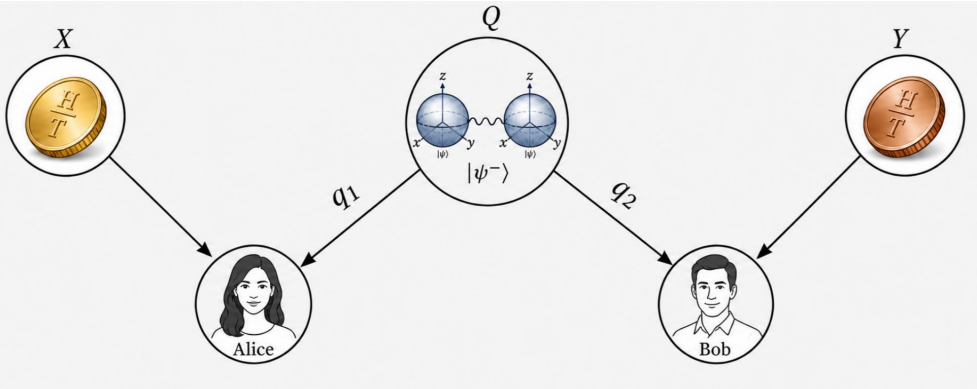
$$\text{quantum axiom} \;\mapsto\; \text{Q-cpt}: \quad \mathtt{meas}_m(\vec{r}) : X \;\mapsto\; \big(x \to \mathcal{J}(E_x^m)\big), \qquad \mathtt{chan}_c(\vec{r}) : \bigotimes_j q'_j \;\mapsto\; \mathcal{J}(\mathcal{E}_c)$$

$$\text{derivation } \pi \;\mapsto\; [\![\pi]\!] \triangleq \sum_{\mathrm{Nm}(\Phi_\pi)-\mathrm{Nm}(J)} \bigodot_{\phi \in \mathrm{Cpts}(\pi)} \phi \qquad \text{(classical names marginalised, qubits traced out)}$$

- Each quantum axiom is a Q-cpt, the Choi image of its measurement or channel; a derivation is the product of its Q-cpts with the names absent from the conclusion summed out. Target: compositionality, as for the classical calculus.

Open questions

- Variable elimination and message passing over Q-factors: elimination orders, width, cost.
- Conditioning and Bayes update on classical evidence; case studies: quantum automata, QKD, stabilizer codes (postselection).



$$Pr(A, B|X, Y)$$

but also

$$St(q_1|B, Y)$$

Summary

- **Bayesian network** = DAG + CPTs; the joint is their product.
- **Inference** = product and sum-out on factors; distributivity lets variable elimination and message passing run in time exponential in the width, not in the number of variables.
- **Higher order** = a probabilistic λ-calculus with ! and der; every ground program compiles to a Bayesian network, and the type says which.
- **Quantum Bayesian network** = DAG + Q-cpts; the same two operations on Q-factors, which contain both factors and tensor networks.
- **Open:** inference algorithms, and a quantum language that compiles to QBNs.
- **Open: Bayesian inference on quantum states**

Thank you.

Discord Server!

We put up a discord server!

- Food recommendation
- Gathering places
- Literature pointers
- Anything else!

