UNIVERSITY OF TORONTO
SCHOOL OF CONTINUING STUDIES

3547 Intelligent Agents & Reinforcement Learning

Module 5: Probabilistic Reasoning

Course Plan

Module Titles
Module 1 – Introduction to Intelligent Agents
Module 2 – Search
Module 3 – Logical Inference
Module 4 – Planning and Knowledge Representation
Module 5 – **Current Focus: Probabilistic Reasoning**
Module 6 – Intro to Reinforcement Learning and Finite Markov Decision Processes
Module 7 – Dynamic Programming and Monte Carlo Methods
Module 8 – Temporal Difference Learning
Module 9 – Function Approximation for RL
Module 10 – Deep Reinforcement Learning and Policy Gradient Methods
Module 11 – Introduction to Advanced DRL
Module 12 – Presentations (no content)

Learning Outcomes for this Module

- Refresh your knowledge of Bayesian probability
- Enable your agents to use probabilistic reasoning as an extension to purely logical reasoning
- Enable you to design probabilistic graphical models
- Give you the skills to begin to use probabilistic programming as a new technique

Topics for this Module

- **5.1** Belief as Probability
- **5.2** Probabilistic Programming
- **5.3** Resources and Wrap-up

Module 5 – Section 1

Belief as Probability

Belief States

- Our Wumpus hunter can do much better if it can take calculated risks
- We will adopt a Bayesian mindset: probability is subjective (i.e. may be different for different people, but certainly not arbitrary)
- We will consider probability a **belief state**: a measure of how much or little the agent knows about something, not a measure of something in the real world
- For our purposes, propositions are still either true or false (and nothing in between)
- Another way to think of probability: as the summarization of (a potentially huge amount of) information

Belief as Probability

- How should we define this quantify of belief in a proposition being true mathematically? Answer: As the sum of the probabilities of the possible worlds in which it is true.
- $\sum_{\omega \in \Omega} P(\omega) = 1$
- $P(\alpha) = \sum_{\omega : \alpha \text{ is true in } \omega} P(\omega)$ for proposition α

- Axioms
  - $0 \leq P(\alpha)$ for any proposition α
  - $P(\tau) = 1$ if τ is true in all possible worlds
  - $P(\alpha \vee \beta) = P(\alpha) + P(\beta)$ if propositions α and β are mutually exclusive (contradictory)

Useful Identities

- $P(\neg\alpha) = 1 - P(\alpha)$
- If $\alpha \leftrightarrow \beta$, then $P(a) = P(\beta)$
- $P(\alpha) = P(\alpha \wedge \beta) + P(\alpha \wedge \neg\beta)$
- $P(\alpha) = \sum_{d \in D} P(\alpha \wedge V = d)$ if V is a random variable with domain d and α is any proposition
- $P(\alpha \vee \beta) = P(\alpha) + P(\beta) - P(\alpha \wedge \beta)$

Conditional Probability

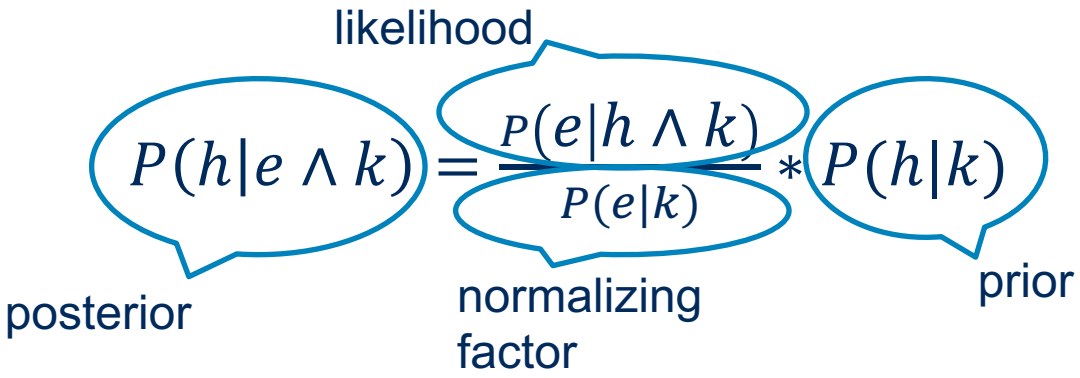
- All probabilities are effectively conditional
- We're interested in how much belief an agent should have in the truth of each proposition given *everything* (relevant) it already knows
- We can describe everything known as a conjunction of all the agent's observations of the environment called the **evidence** e
- We write this **conditional probability** $P(h|e)$ where h is any proposition and e is the evidence
- The initial probability $P(h)$, before the agent has observed anything is called the **prior probability** of h and is the same as $P(h|true)$

Conditional Probability

- Evidence will rule out more and more possible worlds as it is gathered
- What is the conditional probability of world ω given evidence e?
- $P(\omega|e) = \begin{cases} c * P(\omega) \; if \; e \; is \; true \; in \; world \; \omega \\ 0 \; if \; e \; is \; false \; in \; world \; \omega \end{cases}$

Bayes Rule

- Bayes Rule tells us how to update a belief in a hypothesis given new evidence
- The probability of a proposition h being true given we already know k and just found out e is

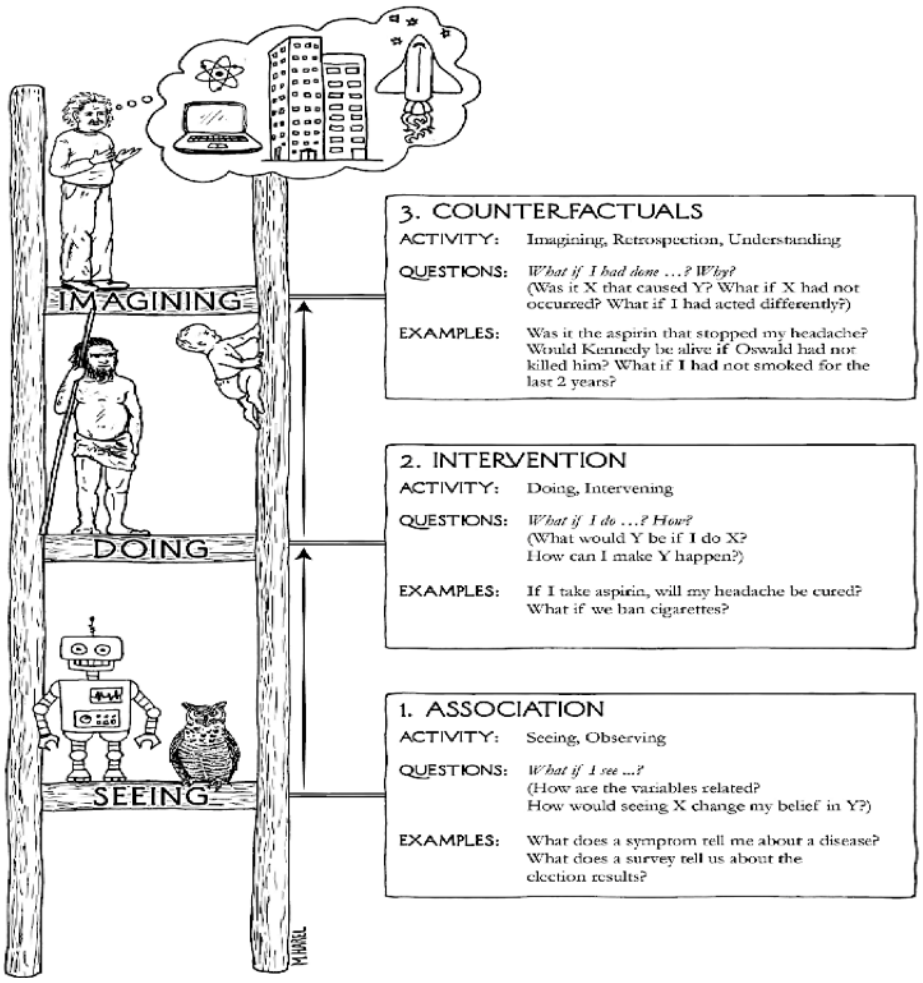


UNIVERSITY OF TORONTO
SCHOOL OF CONTINUING STUDIES

Module 5 – Section 2

Probabilistic Programming

Pearl's Ladder of Causation



3. COUNTERFACTUALS

ACTIVITY: Imagining, Retrospection, Understanding

QUESTIONS: *What if I had done ...? Why?* (Was it X that caused Y? What if X had not occurred? What if I had acted differently?)

EXAMPLES: Was it the aspirin that stopped my headache? Would Kennedy be alive if Oswald had not killed him? What if I had not smoked for the last 2 years?

2. INTERVENTION

ACTIVITY: Doing, Intervening

QUESTIONS: *What if I do ...? How?* (What would Y be if I do X? How can I make Y happen?)

EXAMPLES: If I take aspirin, will my headache be cured? What if we ban cigarettes?

1. ASSOCIATION

ACTIVITY: Seeing, Observing

QUESTIONS: *What if I see ...?* (How are the variables related? How would seeing X change my belief in Y?)

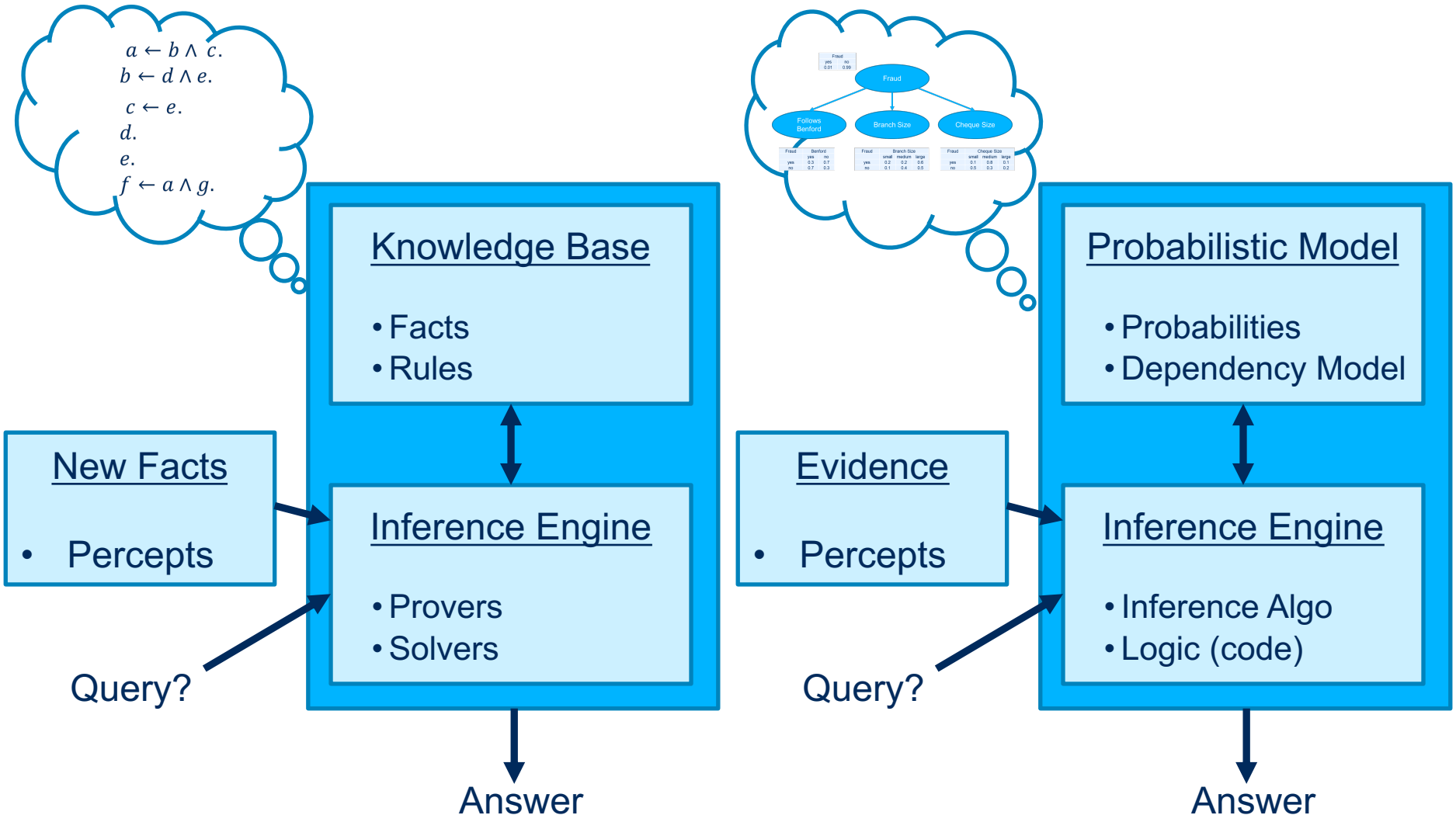
EXAMPLES: What does a symptom tell me about a disease? What does a survey tell us about the election results?

Source: Maayan Harell

What is Probabilistic Programming?

- A new approach to probabilistic reasoning
- Combines the power of probabilistic models with general-purpose (Turing Complete) programming languages
- Examples:
  - STAN (R)
  - PyMC (Python + TensorFlow Probability)
  - Figaro, Rainier (Scala)
  - Gen (Julia)
  - Probabilistic-C
  - WebPPL (JavaScript)

Logic Programming vs. Probabilistic Programming



General Prior Knowledge

- Example: Consider these general facts
  - Numbers on legitimate cheques tend to follow Benford's distribution
  - Fraudulent cheques need to be large enough to be worthwhile but not so large to draw attention
  - Some fraudsters prefer to go to very large branches where they won't be remembered
  - Some fraudsters prefer to go to small, remote branches where the staff may not be as sophisticated

Possibilities for a Specific Cheque

- The cheque does or does not follow Benford's distribution
- It was for a small, medium or large amount
- It was cashed at a small, medium or large branch
- Whether it is fraudulent or not

A Probabilistic Model With Prior Probabilities

World: Fraudulent?, Follows Benford?, Cheque Size?, Branch Size?	Probability of the World
fraud, benford, small, small	0.00006
fraud, benford, small, medium	0.00048
fraud, benford, small, large	0.00006
fraud, benford, medium, small	0.00006
fraud, benford, medium, medium	0.00048
fraud, benford, medium, large	0.00006
fraud, benford, large, small	0.00018
fraud, benford, large, medium	0.00144
fraud, benford, large, large	0.00018
fraud, ¬benford, small, small	0.00014
…	…

$$\sum p = 1$$

Marginal Probabilities

- The probability of any specific fact can be found by summing up the probabilities of all the worlds it is true in
- Examples:
  - $P(fraud) = \sum all\ worlds\ where\ fraud = true$
  - $P(cheque\ cashed\ at\ small\ bank) = \sum all\ worlds\ where\ bank\ size = small$

Using Evidence

- Let's say we have this evidence for a specific cheque:
  - Cheque is for a large amount
  - Its digits follow the Benford distribution
  - The depositor wrote the amount in pencil
- $P(fraud|benford, large\ amount)$
- To use the evidence we remove all worlds that are inconsistent with the facts from our set of possible worlds, then renormalize
- The result is our **posterior probability distribution**

Reduce to Still-Possible Worlds

World: Fraudulent?, Follows Benford?, Cheque Size?, Branch Size?	Probability of the World
~~fraud, benford, small, small~~	~~0.00006~~
~~fraud, benford, small, medium~~	~~0.00048~~
~~fraud, benford, small, large~~	~~0.00006~~
~~fraud, benford, medium, small~~	~~0.00006~~
~~fraud, benford, medium, medium~~	~~0.00048~~
~~fraud, benford, medium, large~~	~~0.00006~~
fraud, benford, large, small	0.00018
fraud, benford, large, medium	0.00144
fraud, benford, large, large	0.00018
~~fraud, ¬benford, small, small~~	~~0.00014~~
…	…

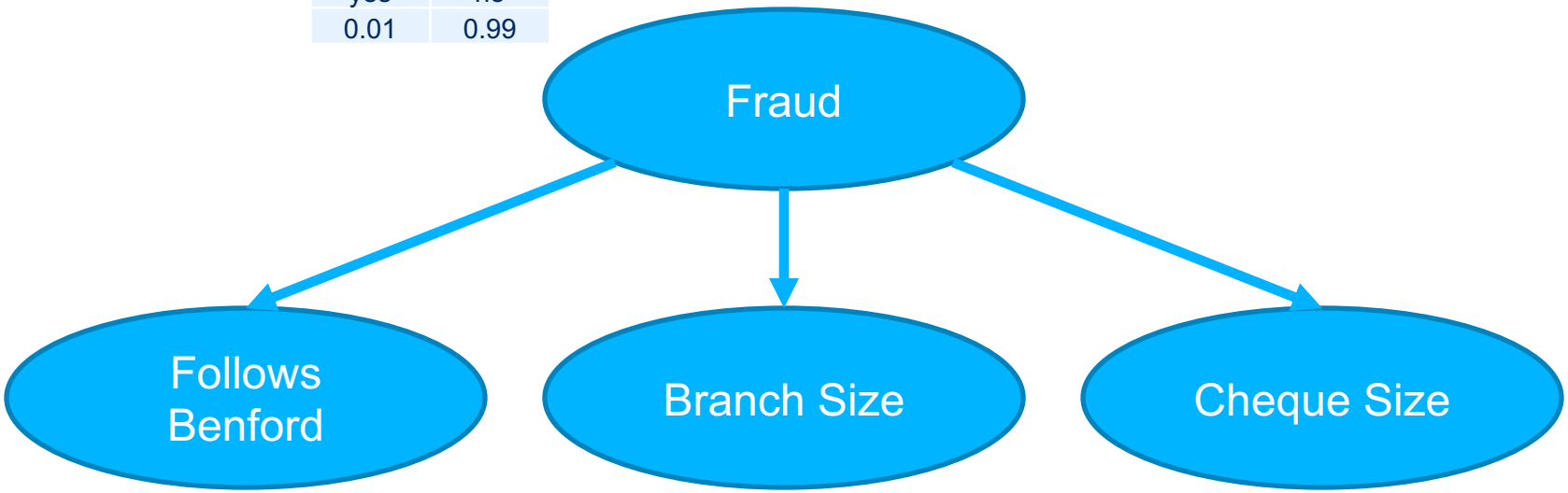
Then renormalize so $\sum p = 1$

Probabilistic Inference

- The goal is to compute the posterior probability distribution over variables of interest
- Since the number of variables can be huge, it may not be possible to calculate the exact probabilities
- Three approaches
  - Chain Rule
  - Total Probability Rule
  - Bayes Rule

Bayesian Network

Fraud	
yes	no
0.01	0.99

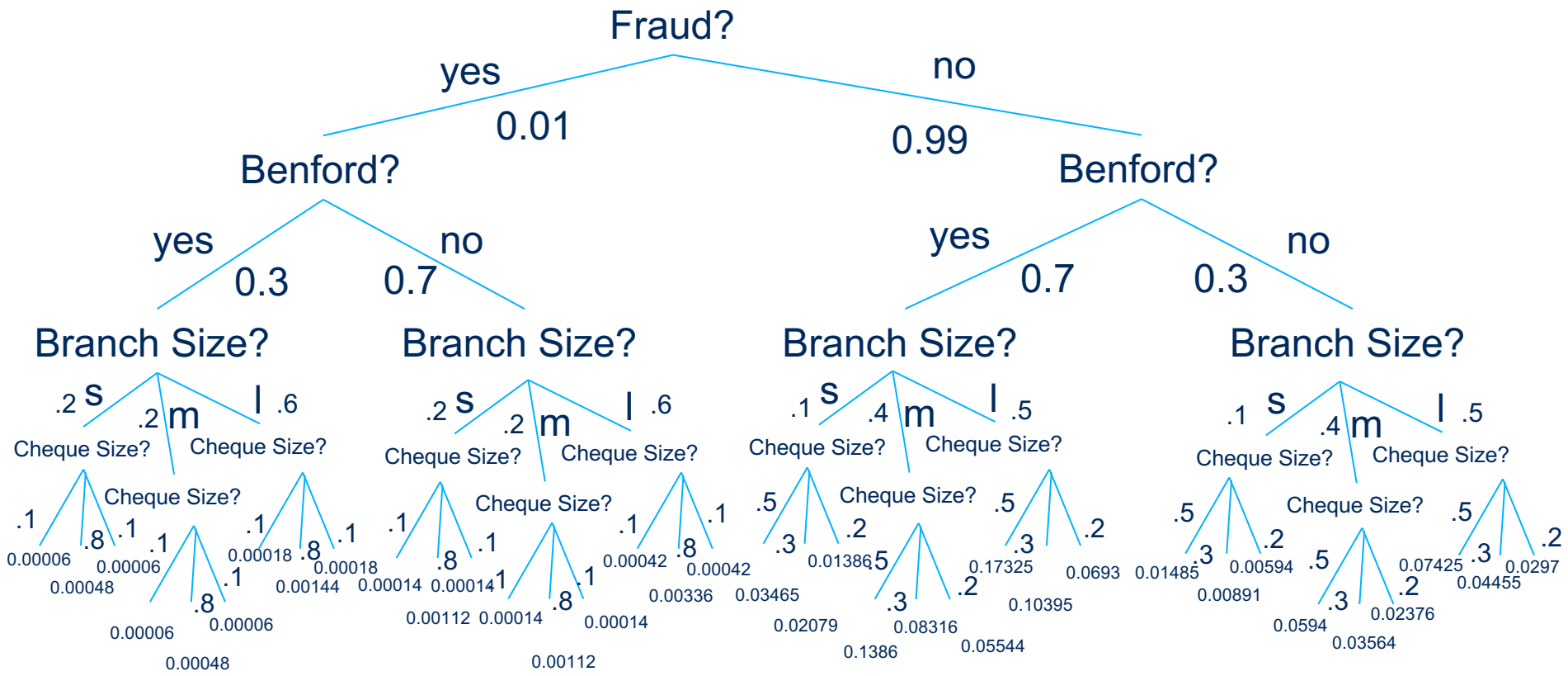


Fraud	Benford	
	yes	no
yes	0.3	0.7
no	0.7	0.3

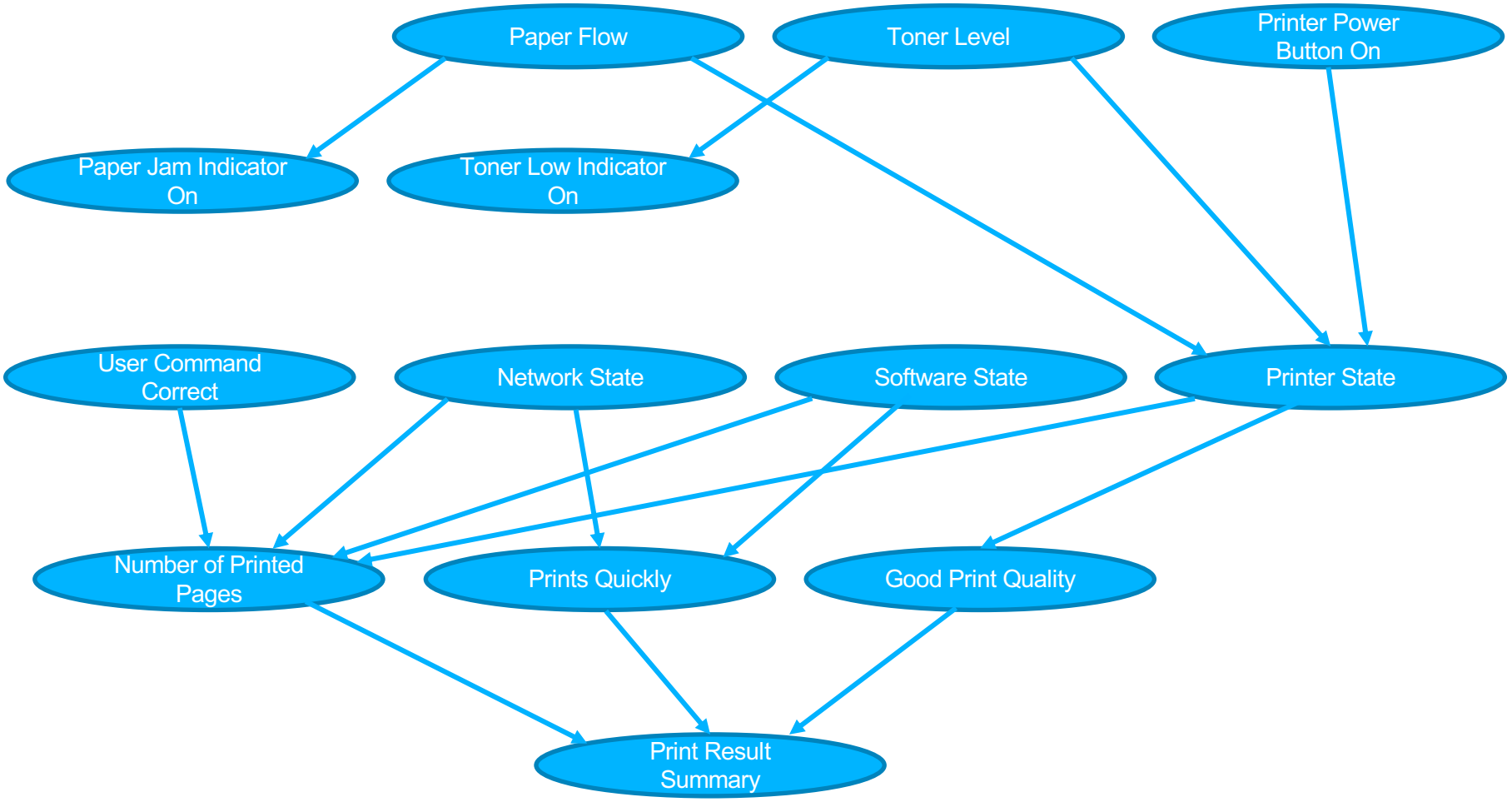
Fraud	Branch Size		
	small	medium	large
yes	0.2	0.2	0.6
no	0.1	0.4	0.5

Fraud	Cheque Size		
	small	medium	large
yes	0.1	0.8	0.1
no	0.5	0.3	0.2

Probabilities of Possible Worlds



A More Complex Example



Source: Pfeffer

Figaro Code

```
val numPrintedPages =
  RichCPD(userCommandCorrect, networkState, softwareState, printerState,
      (*, *, *, OneOf('out)) -> Constant('zero),
      (*, *, OneOf('crashed), *) -> Constant('zero),
      (*, OneOf('down), *, *) -> Constant('zero),
      (OneOf(false), *, *, *) -> Select(0.3 -> 'zero, 0.6 -> 'some, 0.1 -> 'all),
      (OneOf(true), *, *, *) -> Select(0.01 -> 'zero, 0.01 -> 'some, 0.98 -> 'all))
val printsQuickly =
  Chain(networkState, softwareState,
        (network: Symbol, software: Symbol) =>
          if (network == 'down || software == 'crashed) Constant(false)
          else if (network == 'intermittent || software == 'glitchy) Flip(0.5)
          else Flip(0.9))
val goodPrintQuality =
  CPD(printerState,
      'good -> Flip(0.95),
      'poor -> Flip(0.3),
      'out -> Constant(false))
val printResultSummary =
  Apply(numPrintedPages, printsQuickly, goodPrintQuality,
        (pages: Symbol, quickly: Boolean, quality: Boolean) =>
        if (pages == 'zero) 'none
        else if (pages == 'some || !quickly || !quality) 'poor
        else 'excellent)
```

Source: Pfeffer

```
package chap05

import com.cra.figaro.language._
import com.cra.figaro.library.compound._
import com.cra.figaro.algorithm.factored.VariableElimination

object PrinterProblem {
    val printerPowerButtonOn = Flip(0.95)
    val tonerLevel = Select(0.7 -> 'high, 0.2 -> 'low, 0.1 -> 'out)
    val tonerLowIndicatorOn =
      If(printerPowerButtonOn,
         CPD(tonerLevel,
             'high -> Flip(0.2),
             'low -> Flip(0.6),
             'out -> Flip(0.99)),
         Constant(false))
    val paperFlow = Select(0.6 -> 'smooth, 0.2 -> 'uneven, 0.2 -> 'jammed)
    val paperJamIndicatorOn =
      If(printerPowerButtonOn,
         CPD(paperFlow,
             'smooth -> Flip(0.1),
             'uneven -> Flip(0.3),
             'jammed -> Flip(0.99)),
         Constant(false))
    val printerState =
      Apply(printerPowerButtonOn, tonerLevel, paperFlow,
            (power: Boolean, toner: Symbol, paper: Symbol) => {
              if (power) {
                if (toner == 'high && paper == 'smooth) 'good
                else if (toner == 'out || paper == 'out) 'out
                else 'poor
              } else 'out
            })
    val softwareState = Select(0.8 -> 'correct, 0.15 -> 'glitchy, 0.05 -> 'crashed)
    val networkState = Select(0.7 -> 'up, 0.2 -> 'intermittent, 0.1 -> 'down)
    val userCommandCorrect = Flip(0.65)
```

Source: Pfeffer

Result of Running Some Queries

```
Prior probability the printer power button is on = 0.95
Probability the printer power button is on given a poor result = 1.0
Probability the printer power button is on given empty result =
0.8573402523786461
Probability the printer power button is on given out printer state =
0.6551724137931032
Probability the printer power button is on given out printer state and
empty result = 0.655172413793103 3
Prior probability the printer state is good = 0.3989999999999997
Probability printer state is good given low toner indicator =
0.2339832869080779 6
Prior probability the software state is correct = 0.8
Probability software state is correct given network up = 0.8
Probability software state is correct given prints slowly =
0.6197991391678623
Probability software state is correct given prints slowly and network
up = 0.3902439024390243 5
```


Source: Pfeffer

UNIVERSITY OF TORONTO
SCHOOL OF CONTINUING STUDIES

Module 5 – Section 3

Resources and Wrap-up

Resources

- Pearl & Mackenzie. **The Book of Why**. Basic Books. 2018.
- Pearl, Glymour & Jewell. **Causal Inference in Statistics: A Primer**. Wiley. 2016.
- Pearl, Judea. **Theoretical Impediments to Machine Learning With Seven Sparks from the Causal Revolution**: https://arxiv.org/pdf/1801.04016.pdf
- Jaynes, E. T. **Probability Theory: The Logic of Science.** Cambridge Press. 2003.
- Pfeffer, Avi. **Practical Probabilistic Programming**. Manning. 2016. Available for free preview online at: https://www.manning.com/books/practical-probabilistic-programming
- https://en.wikipedia.org/wiki/Probabilistic_programming

Resources (Cont'd)

- Causal Bayesian Networks: https://deepmind.com/blog/article/Causal_Bayesian_Networks
- Koller & Friedman. **Probabilistic Graphical Models.** MIT Press. 2009
- Goodman and Stuhlmüller. **The Design and Implementation of Probabilistic Programming Languages**. http://dippl.org/

Resources (Cont'd)

- TensorFlow Probability: https://www.tensorflow.org/probability
- Gen language: https://probcomp.github.io/Gen/
- Probabilistic C: http://www.robots.ox.ac.uk/~brooks/probabilistic-c/
- Rainier: https://rainier.fit/
- WebPPL: http://webppl.org/
- Python PGM libraries
  - http://pgmpy.org/
  - https://pomegranate.readthedocs.io/en/latest/
- https://pypi.org/project/pymc4/

Summary

- Probabilistic Programming enables probabilistic reasoning, which allows agents to take calculated risks
- Causal reasoning is moving from a “forbidden” concept to an essential tool
- It is incredibly powerful, but computationally expensive and difficult to get a large enough data set to have accurate multi-dimensional priors
- Bayesian probabilistic models are of wide application in general statistics, not just AI

Next Week

- Introduction to Reinforcement Learning
- Markov Decision Processes

Any questions?

UNIVERSITY OF TORONTO
SCHOOL OF CONTINUING STUDIES

Thank You

Thank you for choosing the University of Toronto School of Continuing Studies