Resolution Theorem Proving: Propositional Logic

Convert a Statement to CNF (Conjunctive Normal Form)

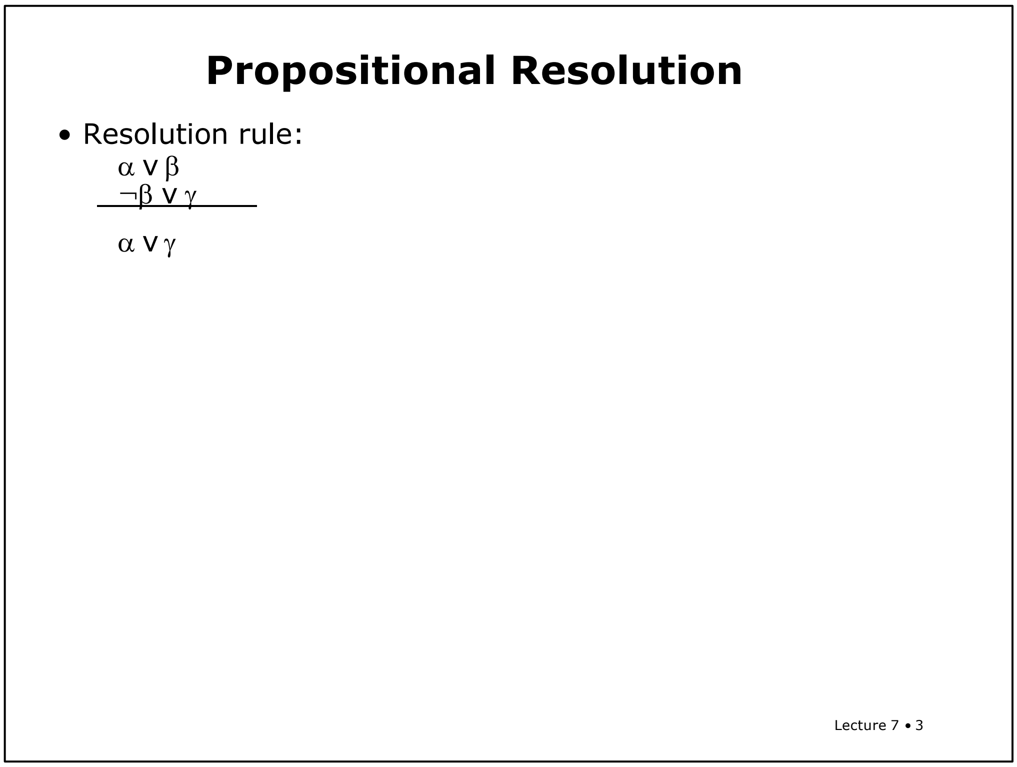
Eliminate implications and bi-conditionals using formulas:

- $(P \Leftrightarrow Q) \implies (P \rightarrow Q) \wedge (Q \rightarrow P)$
- $P \rightarrow Q \Rightarrow \neg P \vee Q$

Resolution

Resolution is a theorem proving technique that proceeds by building refutation proofs, i.e., proofs by contradictions.

Resolution is used, if there are various statements are given, and we need to prove a conclusion of those statements.



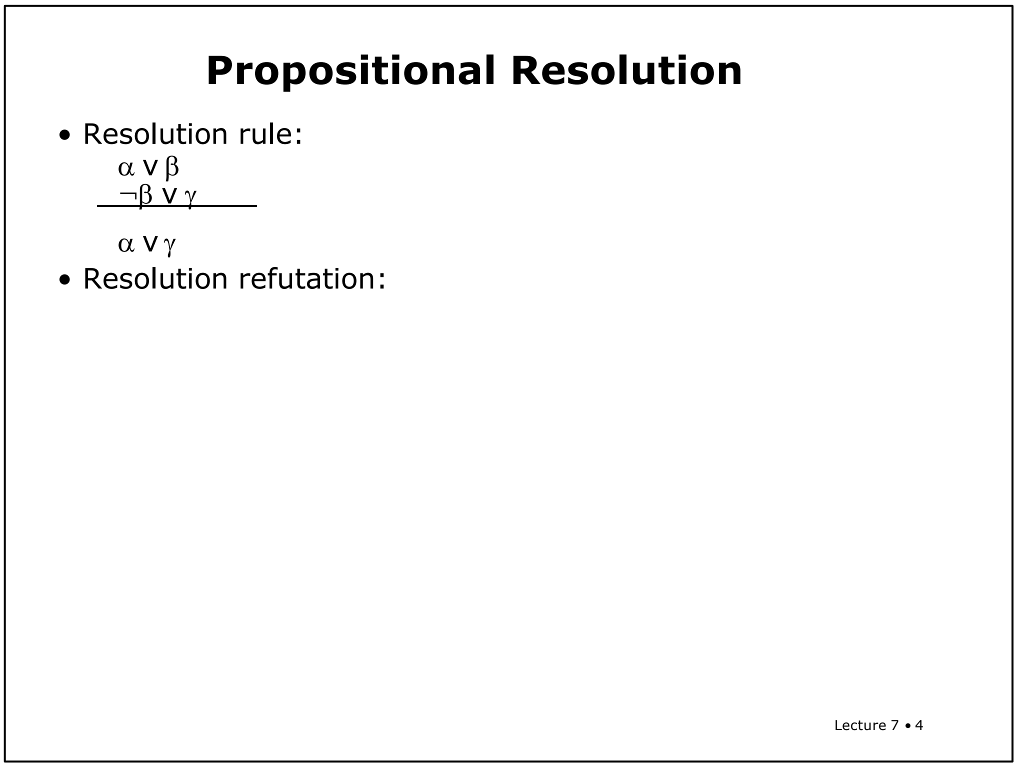
Propositional Resolution

- Resolution rule:

$$\frac{\begin{array}{l}\alpha \vee \beta \\ \neg\beta \vee \gamma\end{array}}{\alpha \vee \gamma}$$

Lecture 7 • 3

So here's the Resolution Inference Rule, in the propositional case. It says that if you know “alpha or beta”, and you know “not beta or gamma”, then you're allowed to conclude “alpha or gamma”. OK. Remember from when we looked at inference rules before that these greek letters are meta-variables. They can stand for big chunks of propositional logic, as long as the parts match up in the right way. So if you know something of the form “alpha or beta”, and you also know that “not beta or gamma”, then you can conclude “alpha or gamma”.



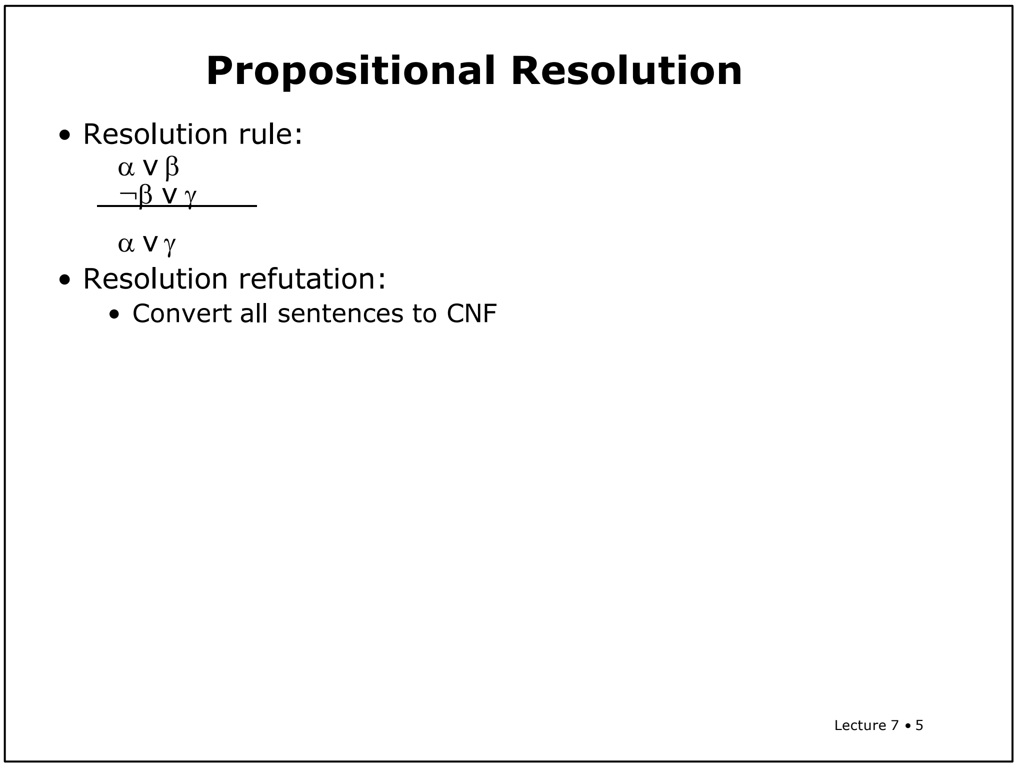
Propositional Resolution

- Resolution rule:

$$\frac{\begin{array}{l}\alpha \vee \beta \\ \neg\beta \vee \gamma\end{array}}{\alpha \vee \gamma}$$

- Resolution refutation:

It turns out that that one rule is all you need to prove things. At least, to prove that a set of sentences is not satisfiable. So, let's see how this is going to work. There's a proof strategy called Resolution Refutation, with three steps. And it goes like this.



Propositional Resolution

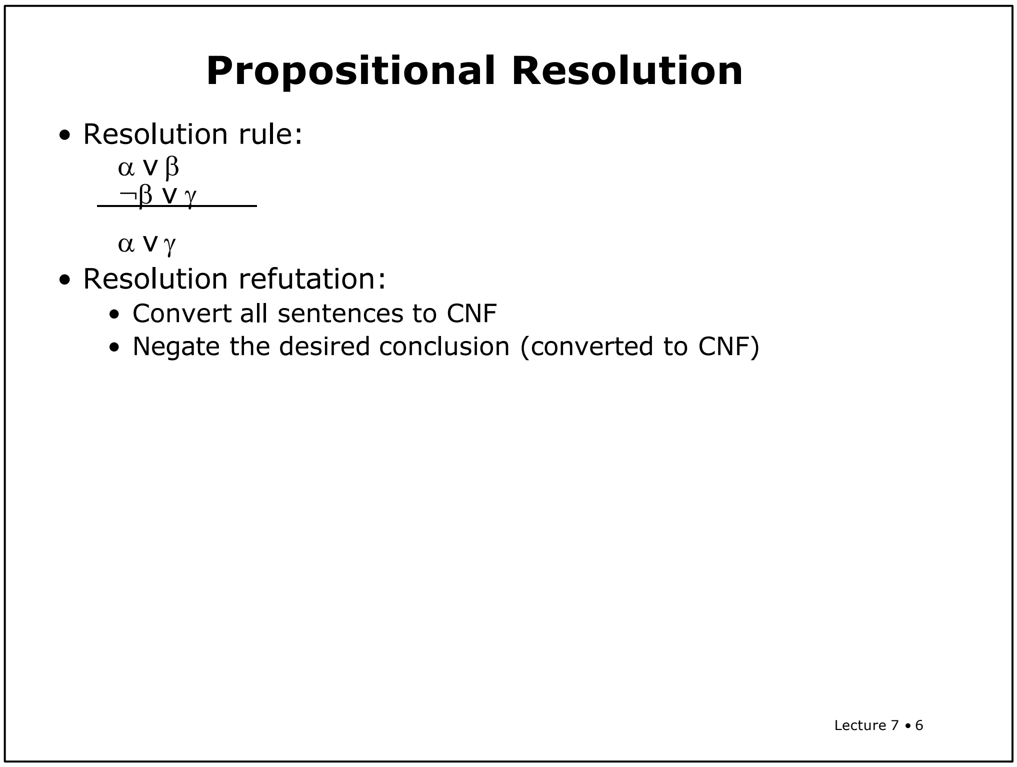
- Resolution rule:

$$\frac{\alpha \vee \beta \quad \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

- Resolution refutation:
  - Convert all sentences to CNF

Lecture 7 • 5

First, you convert all of your sentences to conjunctive normal form. You already know how to do this! Then, you write each clause down as a premise or given in your proof.



Propositional Resolution

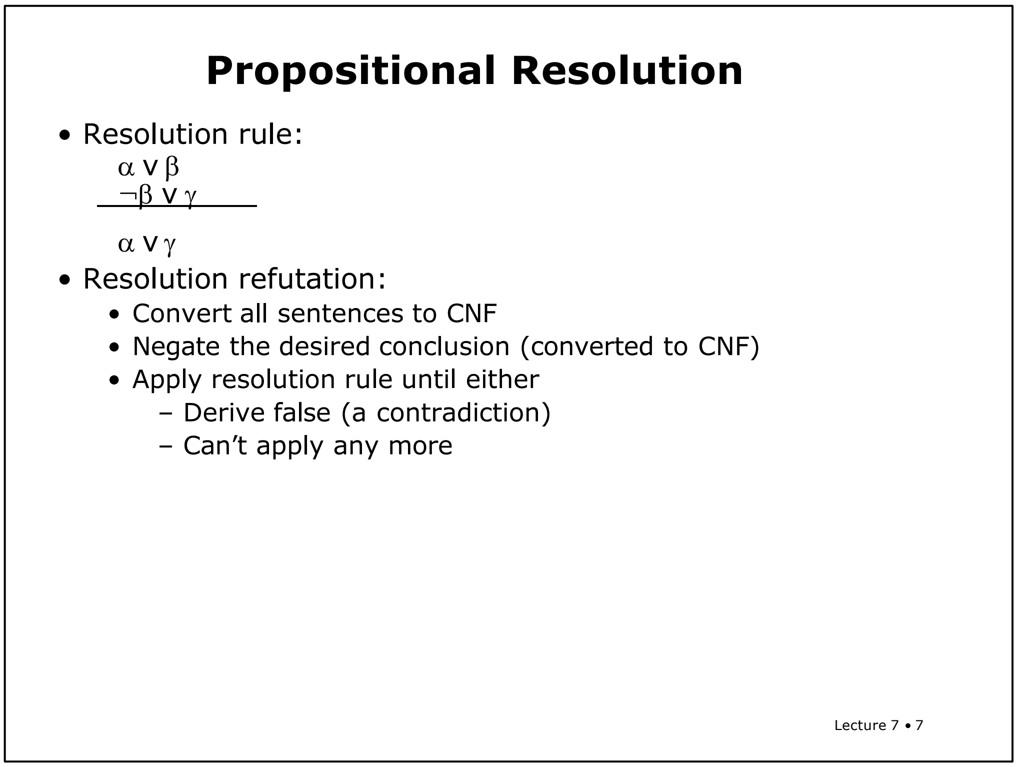
- Resolution rule:

$$\frac{\alpha \vee \beta \qquad \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

- Resolution refutation:
  - Convert all sentences to CNF
  - Negate the desired conclusion (converted to CNF)

Lecture 7 • 6

Then, you negate the desired conclusion -- so you have to say what you're trying to prove, but what we're going to do is essentially a proof by contradiction. You've all seen the strategy of proof by contradiction (or, if we’re being fancy and Latin, *reductio ad absurdum*). You assert that the thing that you're trying to prove is false, and then you try to derive a contradiction. That's what we're going to do. So you negate the desired conclusion and convert that to CNF. And you add each of these clauses as a premise of your proof, as well.



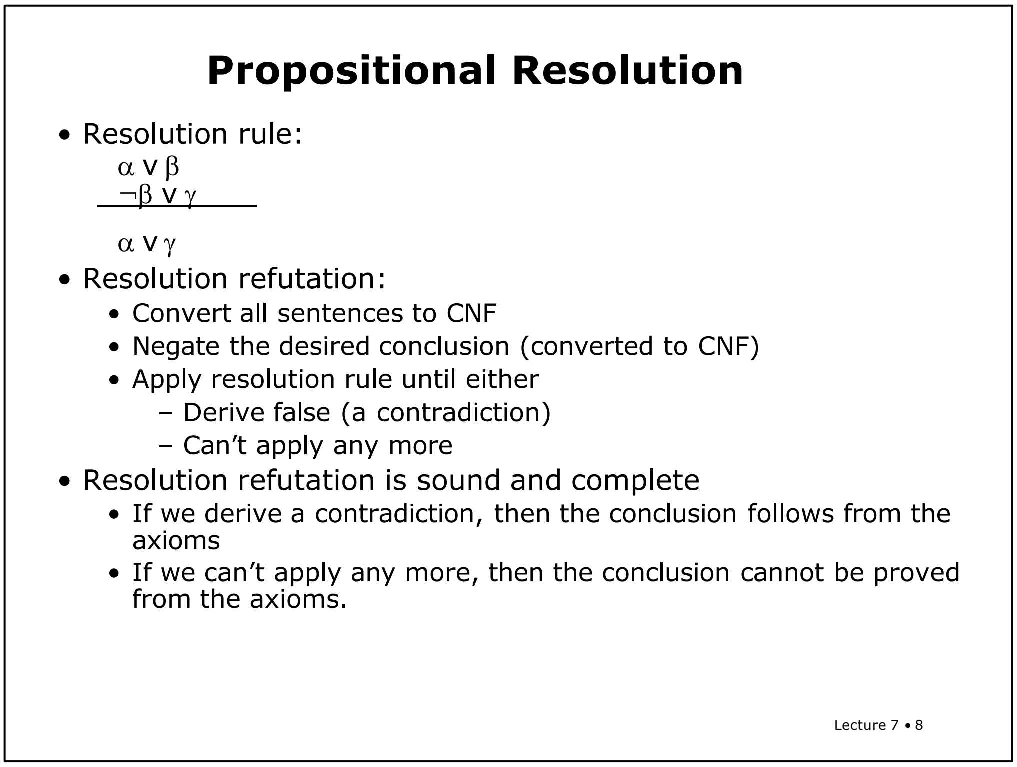
Propositional Resolution

- Resolution rule:

$$\frac{\alpha \vee \beta \qquad \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

- Resolution refutation:
  - Convert all sentences to CNF
  - Negate the desired conclusion (converted to CNF)
  - Apply resolution rule until either
    - Derive false (a contradiction)
    - Can’t apply any more

And then we apply the Resolution Rule until either you can derive "false" -- which means that the conclusion did, in fact, follow from the things that you had assumed, right? If you assert that the negation of the thing that you're interested in is true, and then you prove for a while and you manage to prove false, then you've succeeded in a proof by contradiction of the thing that you were trying to prove in the first place. So you run the resolution rule until you derive false or until you can't apply it anymore.



Propositional Resolution

- Resolution rule:

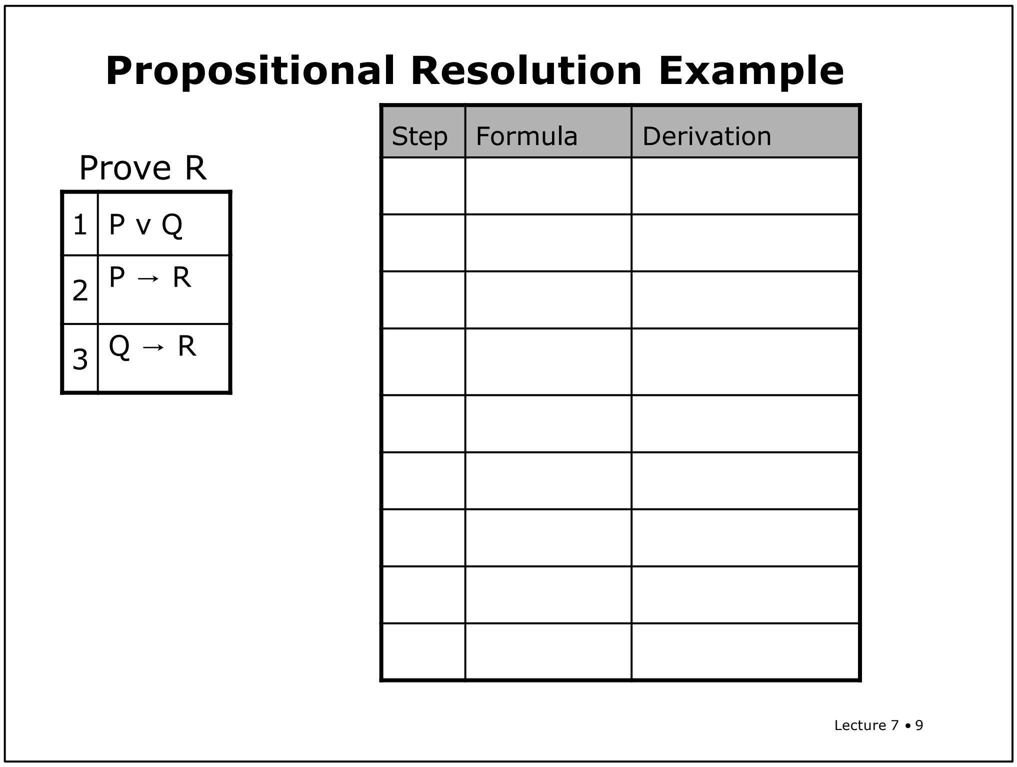
$$\frac{\alpha \vee \beta \qquad \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

- Resolution refutation:
  - Convert all sentences to CNF
  - Negate the desired conclusion (converted to CNF)
  - Apply resolution rule until either
    - Derive false (a contradiction)
    - Can't apply any more
- Resolution refutation is sound and complete
  - If we derive a contradiction, then the conclusion follows from the axioms
  - If we can't apply any more, then the conclusion cannot be proved from the axioms.

Lecture 7 • 8

What if you can't apply the Resolution Rule anymore? Is there anything in particular that you can conclude? In fact, you can conclude that the thing that you were trying to prove can't be proved. So resolution refutation for propositional logic is a complete proof procedure. So if the thing that you're trying to prove is, in fact, entailed by the things that you've assumed, then you can prove it using resolution refutation. In the propositional case. It's more complicated in the first-order case, as we'll see. But in the propositional case, it means that if you've applied the Resolution Rule and you can't apply it anymore, then your desired conclusion can't be proved.

It's guaranteed that you'll always either prove false, or run out of possible steps. It's complete, because it always generates an answer. Furthermore, the process is sound: the answer is always correct.



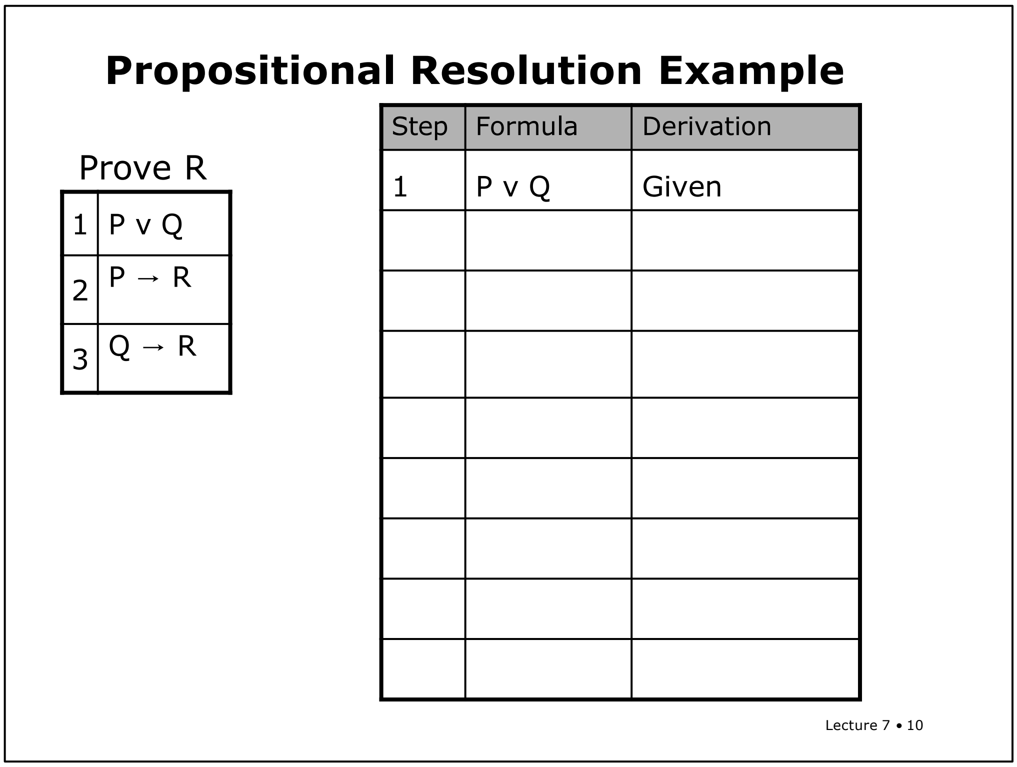
Propositional Resolution Example

Prove R

1	P v Q
2	P → R
3	Q → R

Step	Formula	Derivation

So let's just do a proof. Let's say I'm given “P or Q”, “P implies R” and “Q implies R”. I would like to conclude R from these three axioms. I'll use the word "axiom" just to mean things that are given to me right at the moment.



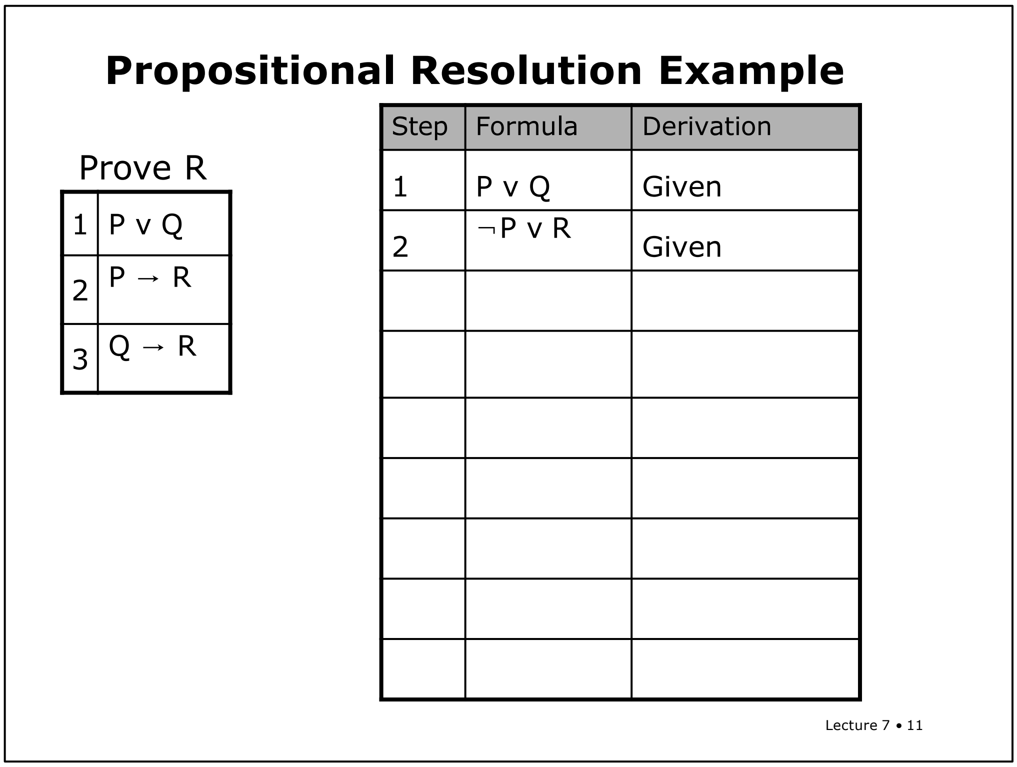
Propositional Resolution Example

Prove R

1	P v Q
2	P → R
3	Q → R

Step	Formula	Derivation
1	P v Q	Given

We start by converting this first sentence into conjunctive normal form. We don't actually have to do anything. It's already in the right form.



Propositional Resolution Example

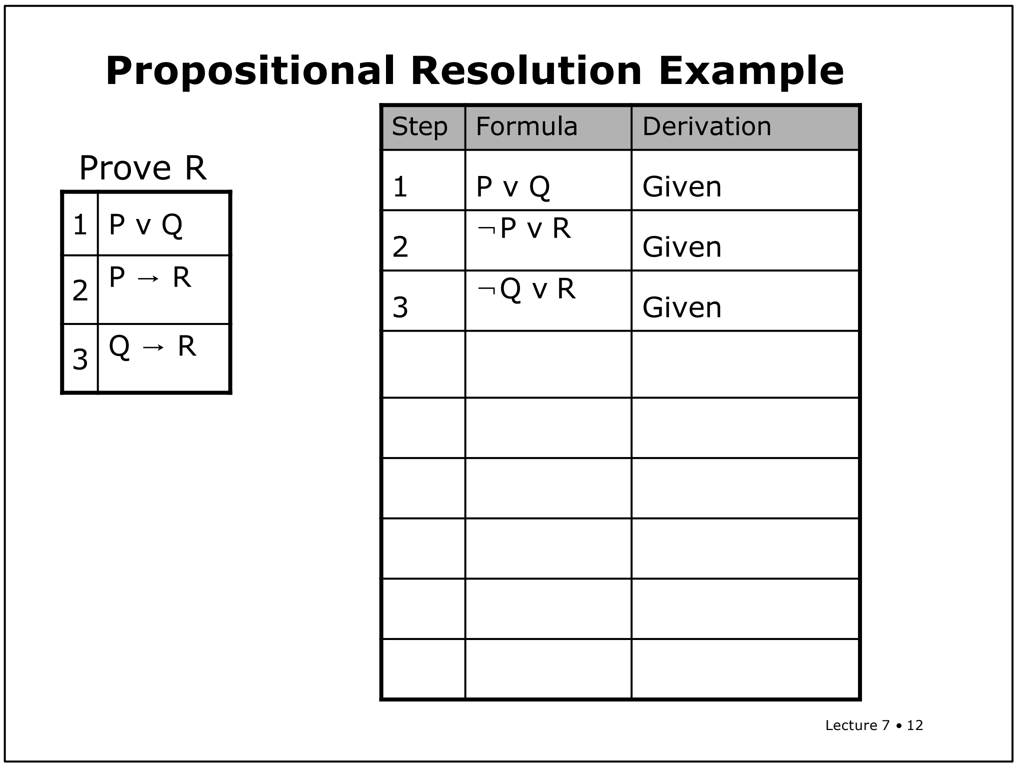
Prove R

1	P v Q
2	P → R
3	Q → R

Step	Formula	Derivation
1	P v Q	Given
2	¬P v R	Given

Lecture 7 • 11

Now, “P implies R” turns into “not P or R”.



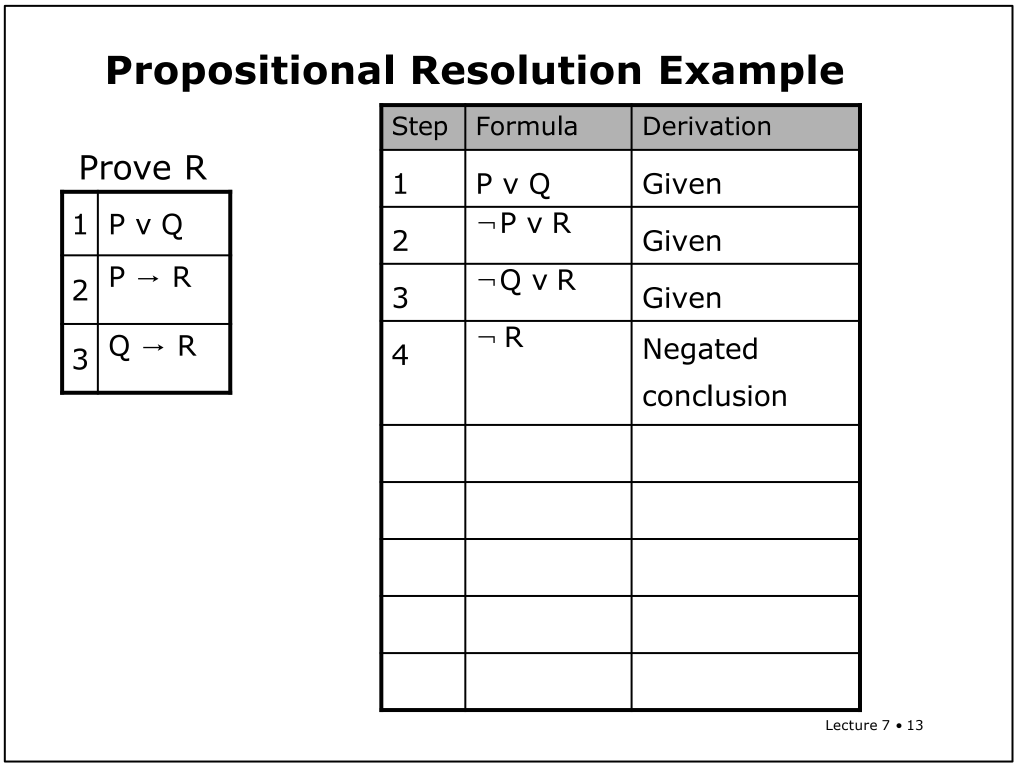
Propositional Resolution Example

Prove R

1	P v Q
2	P → R
3	Q → R

Step	Formula	Derivation
1	P v Q	Given
2	¬P v R	Given
3	¬Q v R	Given

Similarly, “Q implies R” turns into “not Q or R



Propositional Resolution Example

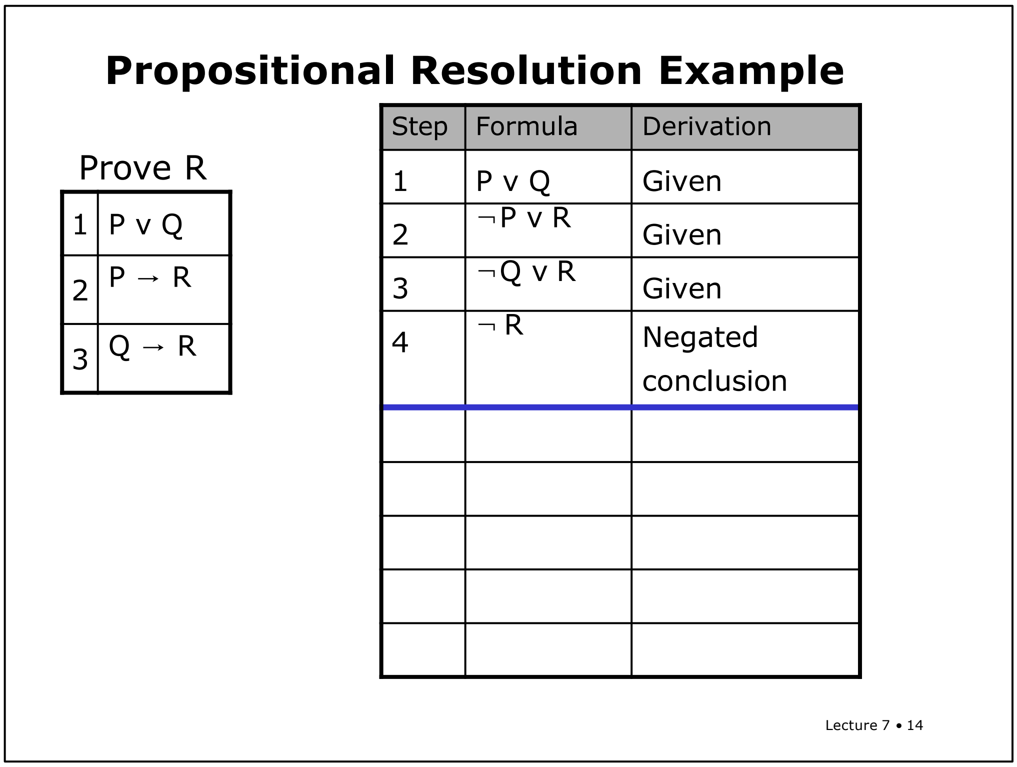
Prove R

1	P v Q
2	P → R
3	Q → R

Step	Formula	Derivation
1	P v Q	Given
2	¬P v R	Given
3	¬Q v R	Given
4	¬ R	Negated conclusion

Now we want to add one more thing to our list of given statements. What's it going to be?

Not R. Right? We're going to assert the negation of the thing we're trying to prove. We'd like to prove that R follows from these things. But what we're going to do instead is say not R, and now we're trying to prove false. And if we manage to prove false, then we will have a proof that R is entailed by the assumptions.



Propositional Resolution Example

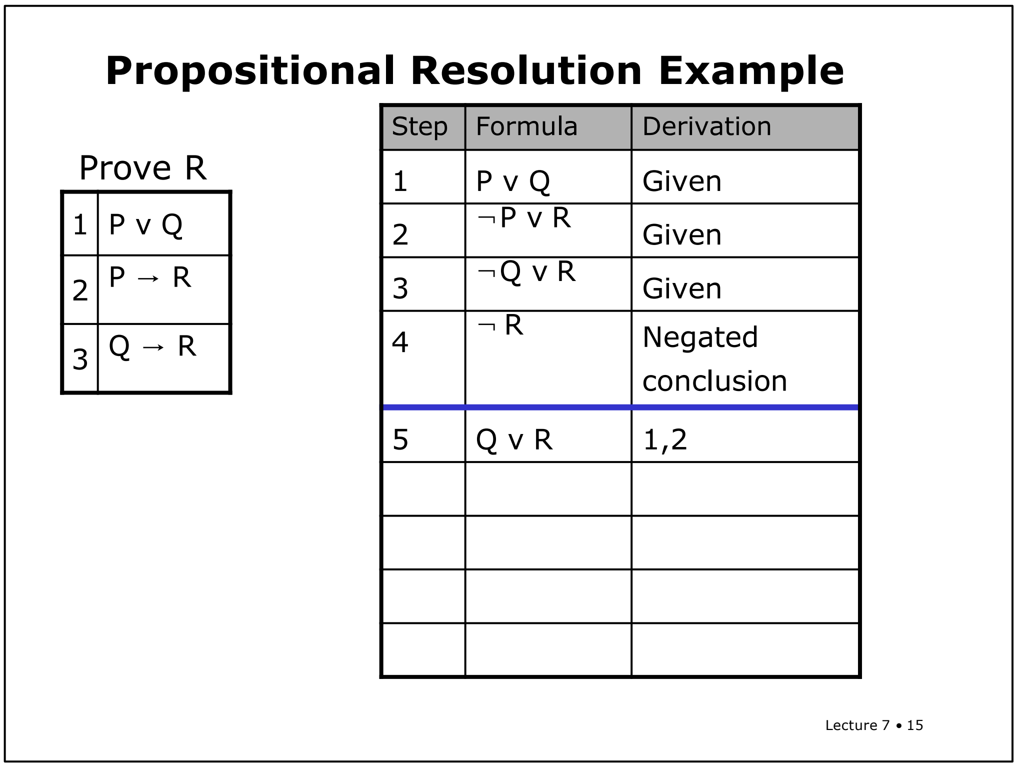
Prove R

1	P v Q
2	P → R
3	Q → R

Step	Formula	Derivation
1	P v Q	Given
2	¬P v R	Given
3	¬Q v R	Given
4	¬ R	Negated conclusion

Now, we’ll draw a blue line just to divide the assumptions from the proof steps.

And now, we look for opportunities to apply the resolution rule. You can do it in any order you like (though some orders of application will result in much shorter proofs than others).



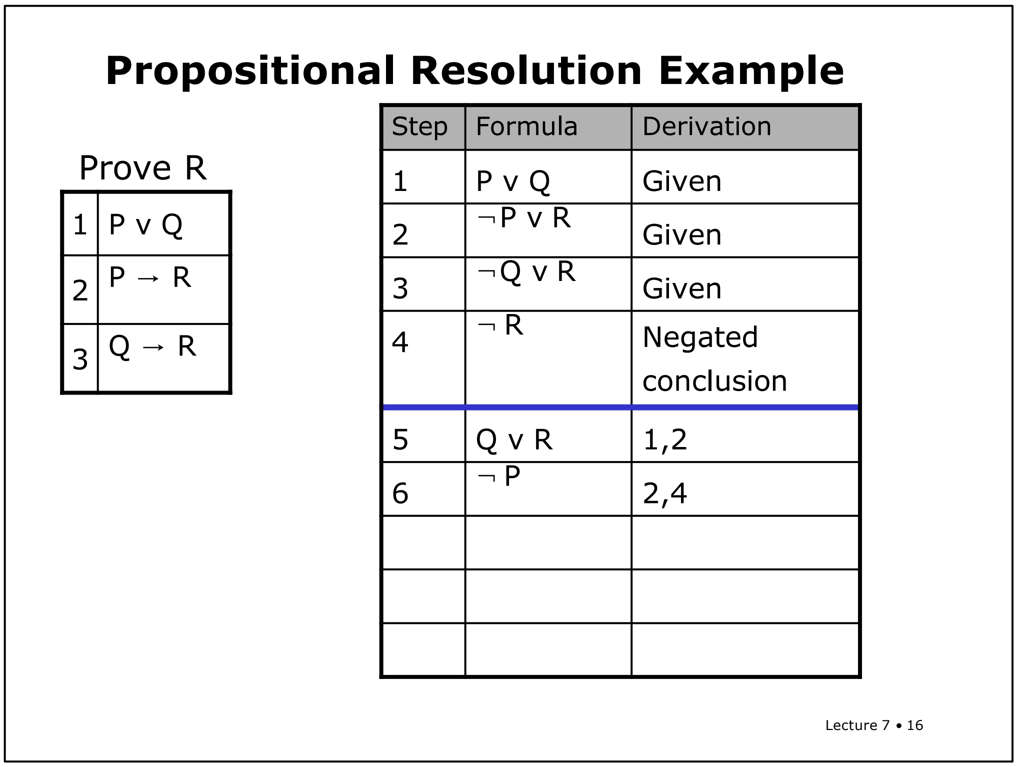
Propositional Resolution Example

Prove R

1	P v Q
2	P → R
3	Q → R

Step	Formula	Derivation
1	P v Q	Given
2	¬P v R	Given
3	¬Q v R	Given
4	¬ R	Negated conclusion
5	Q v R	1,2

We can apply resolution to lines 1 and 2, and get “Q or R” by resolving away P.



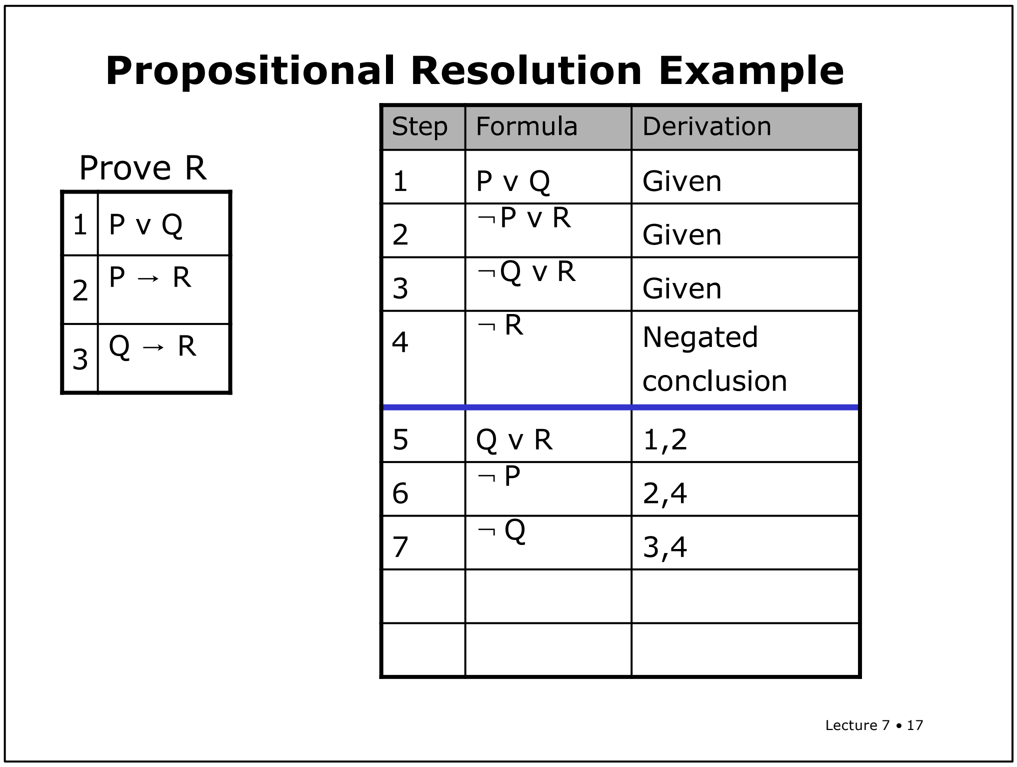
Propositional Resolution Example

Prove R

1	P v Q
2	P → R
3	Q → R

Step	Formula	Derivation
1	P v Q	Given
2	¬P v R	Given
3	¬Q v R	Given
4	¬ R	Negated conclusion
5	Q v R	1,2
6	¬ P	2,4

And we can take lines 2 and 4, resolve away R, and get “not P.”



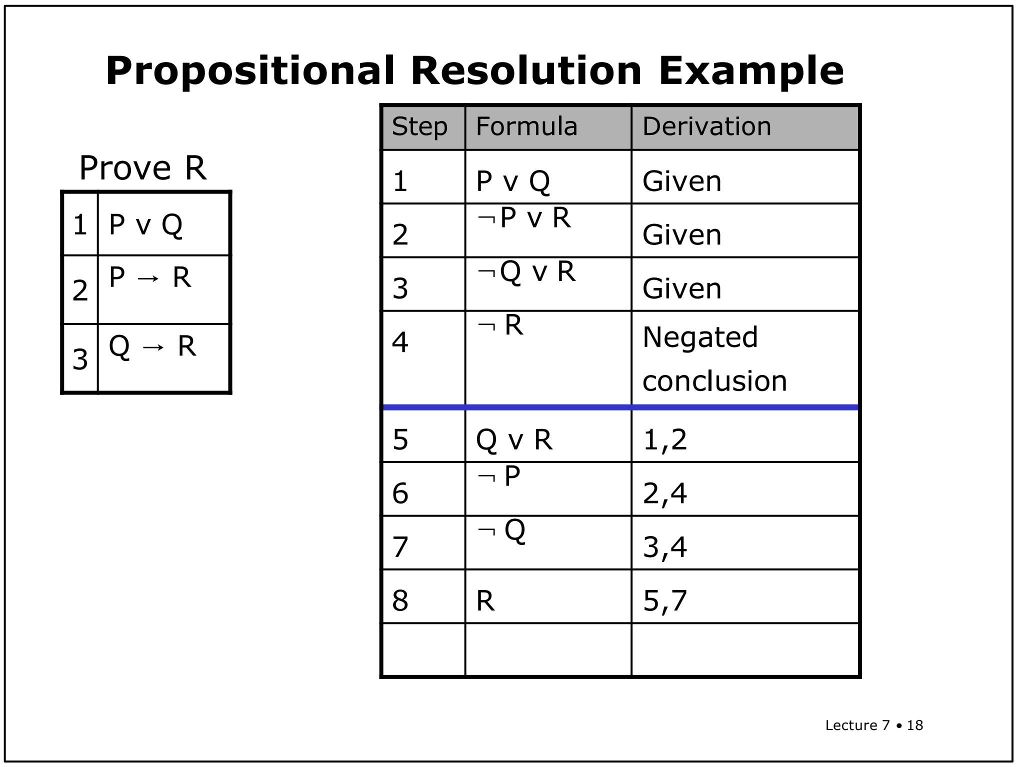
Propositional Resolution Example

Prove R

1	P v Q
2	P → R
3	Q → R

Step	Formula	Derivation
1	P v Q	Given
2	¬P v R	Given
3	¬Q v R	Given
4	¬ R	Negated conclusion
5	Q v R	1,2
6	¬ P	2,4
7	¬ Q	3,4

Similarly, we can take lines 3 and 4, resolve away R, and get “not Q”.



Propositional Resolution Example

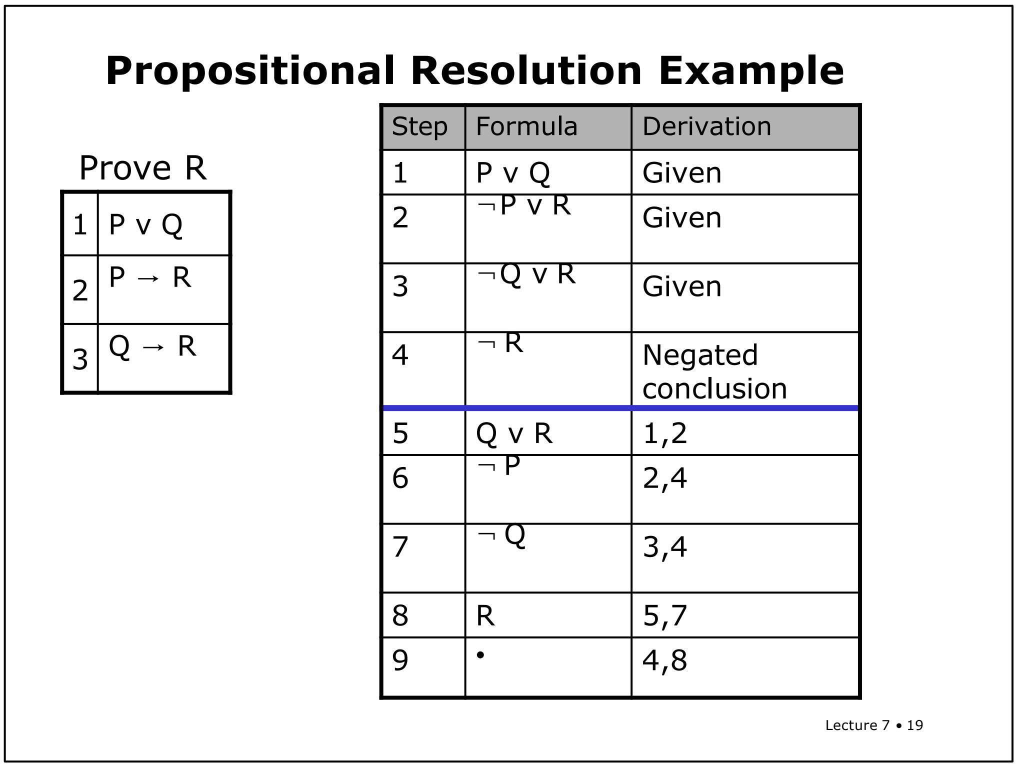
Prove R

1	P v Q
2	P → R
3	Q → R

Step	Formula	Derivation
1	P v Q	Given
2	¬P v R	Given
3	¬Q v R	Given
4	¬ R	Negated conclusion
5	Q v R	1,2
6	¬ P	2,4
7	¬ Q	3,4
8	R	5,7

Lecture 7 • 18

By resolving away Q in lines 5 and 7, we get R.



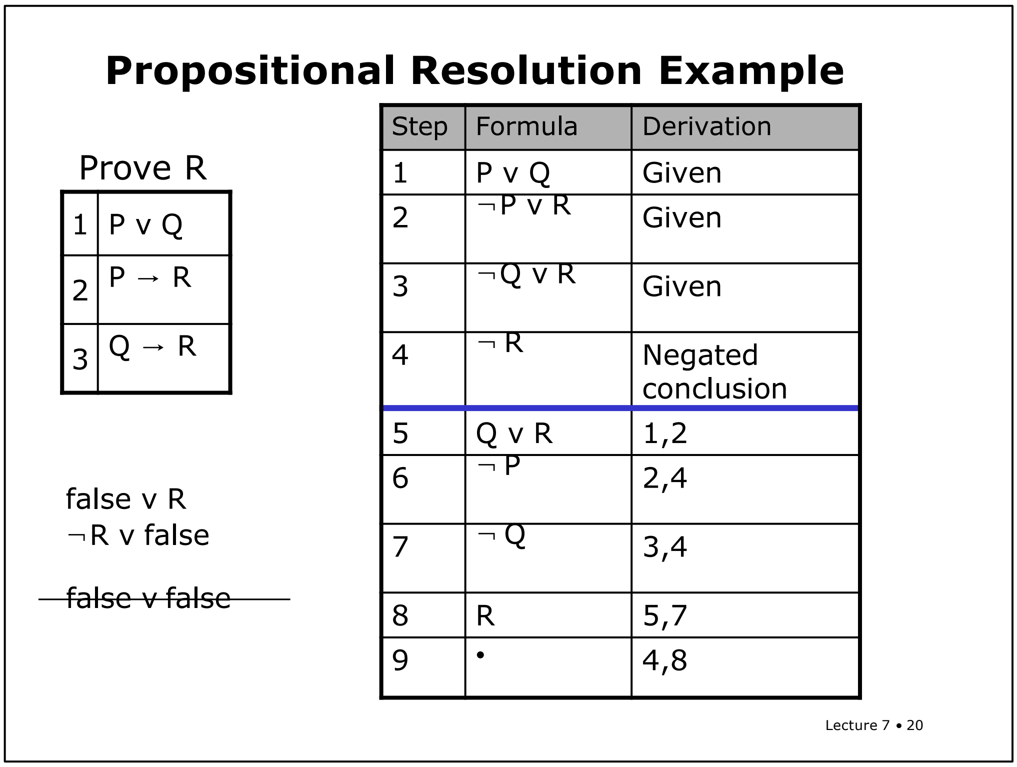
Propositional Resolution Example

Prove R

1	P v Q
2	P → R
3	Q → R

Step	Formula	Derivation
1	P v Q	Given
2	¬P v R	Given
3	¬Q v R	Given
4	¬ R	Negated conclusion
5	Q v R	1,2
6	¬ P	2,4
7	¬ Q	3,4
8	R	5,7
9	•	4,8

And finally, resolving away R in lines 4 and 8, we get the empty clause, which is false. We'll often draw this little black box to indicate that we've reached the desired contradiction.



Propositional Resolution Example

Prove R

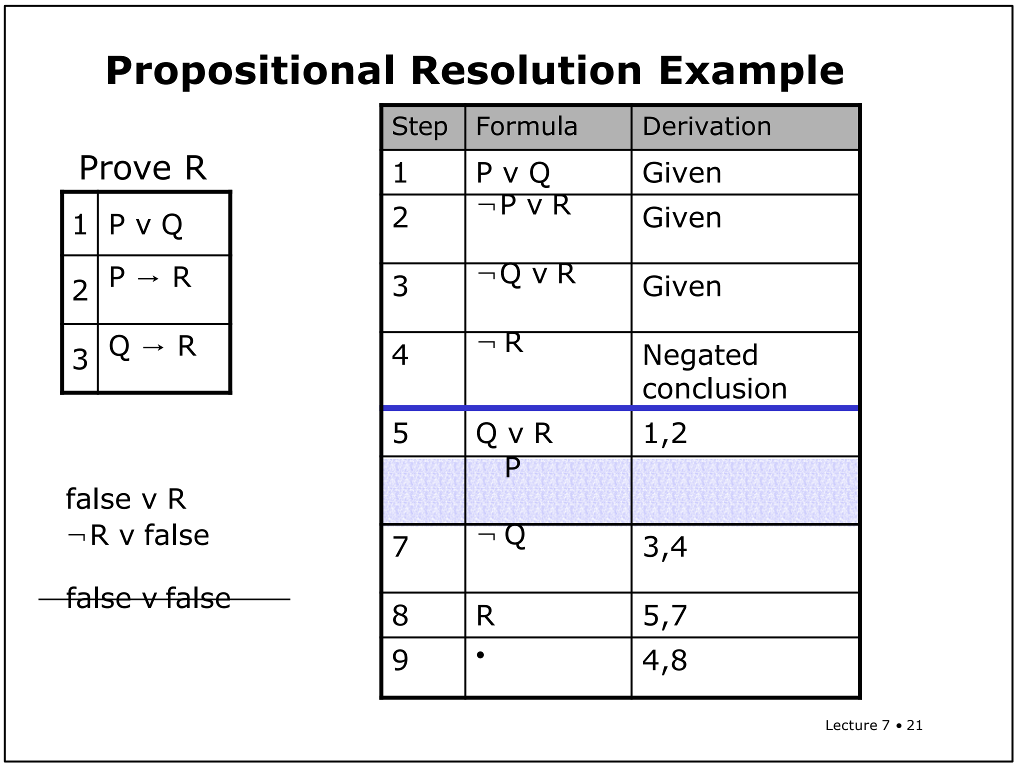
1	P v Q
2	P → R
3	Q → R

false v R
¬R v false

~~false v false~~

Step	Formula	Derivation
1	P v Q	Given
2	¬P v R	Given
3	¬Q v R	Given
4	¬ R	Negated conclusion
5	Q v R	1,2
6	¬ P	2,4
7	¬ Q	3,4
8	R	5,7
9	•	4,8

How did I do this last resolution? Let's see how the resolution rule is applied to lines 4 and 8. The way to look at it is that R is really "false or R", and that "not R" is really "not R or false". (Of course, the order of the disjuncts is irrelevant, because disjunction is commutative). So, now we resolve away R, getting "false or false", which is false.



Propositional Resolution Example

Prove R

1	P v Q
2	P → R
3	Q → R

false v R
¬R v false

~~false v false~~

Step	Formula	Derivation
1	P v Q	Given
2	¬P v R	Given
3	¬Q v R	Given
4	¬ R	Negated conclusion
5	Q v R	1,2
	P	
7	¬ Q	3,4
8	R	5,7
9	•	4,8

Lecture 7 • 21

One of these steps is unnecessary. Which one? Line 6. It's a perfectly good proof step, but it doesn't contribute to the final conclusion, so we could have omitted it.

Steps for Resolution:

1. Conversion of facts into first-order logic.
2. Convert FOL statements into CNF
3. Negate the statement which needs to prove (proof by contradiction)
4. Draw resolution graph (unification).

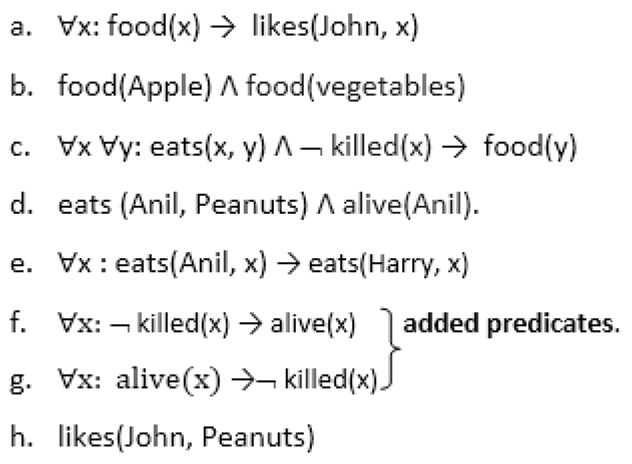
To better understand all the above steps, we will take an example in which we will apply resolution.

Example:

1. **John likes all kind of food.**
2. **Apple and vegetable are food**
3. **Anything anyone eats and not killed is food.**
4. **Anil eats peanuts and still alive**
5. **Harry eats everything that Anil eats.**
   Prove by resolution that:
6. **John likes peanuts.**

Step-1: Conversion of Facts into FOL

In the first step we will convert all the given statements into its first order logic.



a. ∀x: food(x) → likes(John, x)

b. food(Apple) Λ food(vegetables)

c. ∀x ∀y: eats(x, y) Λ ¬ killed(x) → food(y)

d. eats (Anil, Peanuts) Λ alive(Anil).

e. ∀x : eats(Anil, x) → eats(Harry, x)

f. ∀x: ¬ killed(x) → alive(x) } **added predicates.**

g. ∀x: alive(x) →¬ killed(x)

h. likes(John, Peanuts)

Step-2: Conversion of FOL into CNF

In First order logic resolution, it is required to convert the FOL into CNF as CNF form makes easier for resolution proofs.

- **Eliminate all implication (→) and rewrite**
  1. ∀x ¬ food(x) V likes(John, x)
  2. food(Apple) Λ food(vegetables)
  3. ∀x ∀y ¬ [eats(x, y) Λ ¬ killed(x)] V food(y)
  4. eats (Anil, Peanuts) Λ alive(Anil)
  5. ∀x ¬ eats(Anil, x) V eats(Harry, x)

6. ∀x¬ [¬ killed(x)] V alive(x)
7. ∀x ¬ alive(x) V ¬ killed(x)
8. likes(John, Peanuts).

- **Move negation (¬)inwards and rewrite**
  1. ∀x ¬ food(x) V likes(John, x)
  2. food(Apple) Λ food(vegetables)
  3. ∀x ∀y ¬ eats(x, y) V killed(x) V food(y)
  4. eats (Anil, Peanuts) Λ alive(Anil)
  5. ∀x ¬ eats(Anil, x) V eats(Harry, x)
  6. ∀x killed(x) V alive(x)
  7. ∀x ¬ alive(x) V ¬ killed(x)
  8. likes(John, Peanuts).
- **Rename variables or standardize variables**
  1. ∀x ¬ food(x) V likes(John, x)
  2. food(Apple) Λ food(vegetables)
  3. ∀y ∀z ¬ eats(y, z) V killed(y) V food(z)
  4. eats (Anil, Peanuts) Λ alive(Anil)
  5. ∀w¬ eats(Anil, w) V eats(Harry, w)
  6. ∀g killed(g) V alive(g)
  7. ∀k ¬ alive(k) V ¬ killed(k)
  8. likes(John, Peanuts).
- **Eliminate existential instantiation quantifier by elimination.**
  In this step, we will eliminate existential quantifier ∃, and this process is known as **Skolemization**. But in this example problem since there is no existential quantifier so all the statements will remain same in this step.
- **Drop Universal quantifiers.**
  In this step we will drop all universal quantifier since all the statements are not implicitly quantified so we don't need it.
  1. ¬ food(x) V likes(John, x)
  2. food(Apple)
  3. food(vegetables)
  4. ¬ eats(y, z) V killed(y) V food(z)
  5. eats (Anil, Peanuts)
  6. alive(Anil)
  7. ¬ eats(Anil, w) V eats(Harry, w)
  8. killed(g) V alive(g)
  9. ¬ alive(k) V ¬ killed(k)
  10. likes(John, Peanuts).

Note: Statements "food(Apple) Λ food(vegetables)" and "eats (Anil, Peanuts) Λ alive(Anil)" can be written in two separate statements.

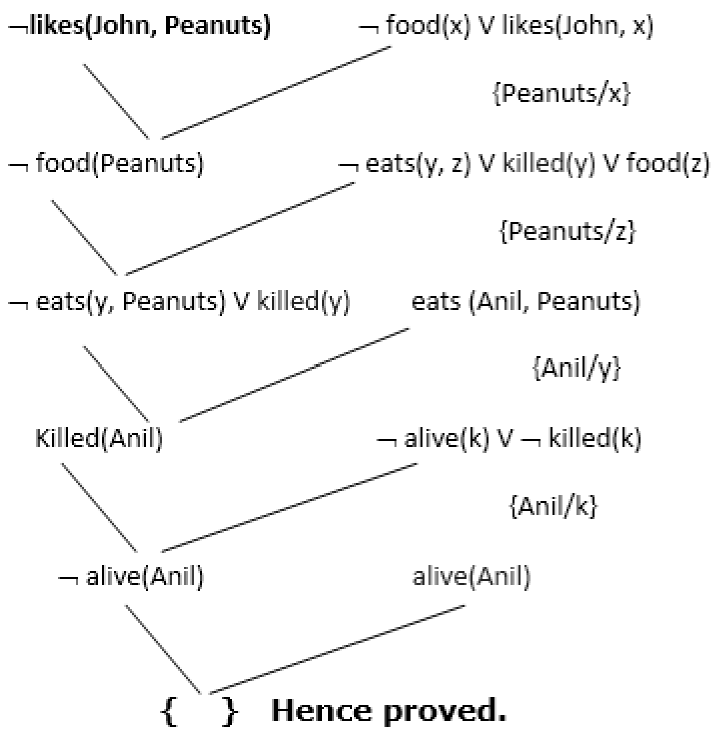
- **Distribute conjunction ∧ over disjunction ¬.**
  This step will not make any change in this problem.

Step-3: Negate the statement to be proved

In this statement, we will apply negation to the conclusion statements, which will be written as ¬likes(John, Peanuts)

Step-4: Draw Resolution graph:

Now in this step, we will solve the problem by resolution tree using substitution. For the above problem, it will be given as follows:



Hence the negation of the conclusion has been proved as a complete contradiction with the given set of statements.

Explanation of Resolution graph:

- In the first step of resolution graph, **¬likes(John, Peanuts)** , and **likes(John, x)** get resolved(canceled) by substitution of **{Peanuts/x}**, and we are left with **¬ food(Peanuts)**
- In the second step of the resolution graph, **¬ food(Peanuts)** , and **food(z)** get resolved (canceled) by substitution of **{ Peanuts/z}**, and we are left with **¬ eats(y, Peanuts) V killed(y)** .
- In the third step of the resolution graph, **¬ eats(y, Peanuts)** and **eats (Anil, Peanuts)** get resolved by substitution **{Anil/y}**, and we are left with **Killed(Anil)** .
- In the fourth step of the resolution graph, **Killed(Anil)** and **¬ killed(k)** get resolve by substitution **{Anil/k}**, and we are left with **¬ alive(Anil)** .
- In the last step of the resolution graph **¬ alive(Anil)** and **alive(Anil)** get resolved.

Reasoning Under Uncertainty

- Introduction
- Representing uncertain knowledge: logic and probability (a reminder!)
- Probabilistic inference using the joint probability distribution
- Bayesian networks (theory and algorithms)

The Importance of Uncertainty

Uncertainty is unavoidable in everyday reasoning and in many real-world domains.

Examples:

- Waiting for a colleague who has not shown up for a meeting.
- Deciding whether to go to school on a very snowy winter morning.
- Judgmental domains such as medicine, business, law and so on.

Sources of Uncertainty

- **Incomplete knowledge.** E.g., laboratory data can be late, medical science can have an incomplete theory for some diseases.
- **Imprecise knowledge.** E.g., the time that an event happened can be known only approximately.
- **Unreliable knowledge.** E.g., a measuring instrument can be biased or defective.

Representing Certain Knowledge: Logic

Example: Patient John has a cavity.

How can we represent this fact in logic?

- Propositional logic: $Cavity$
- First-order logic: $DentalDisease(John, Cavity)$

Ontological commitments: Facts hold or do not hold in the world.

Epistemological commitments: An agent believes a sentence to be true, false or has no opinion.

' $

Question

How can an agent capture the fact that he is **not certain** that John has a cavity?

& %

First Answer

Use logic:

I have no knowledge regarding whether John has a cavity or not.

The formulas $Cavity$ and $\neg Cavity$ do not follow from my KB.

Second (Better?) Answer

Use probabilities:

The probability that patient John has a cavity is 0.8.

We might know this from statistical data or some general dental knowledge.

Representing Uncertain Knowledge: Probability

- Probabilities provide us with a way of assigning **degrees of belief** in a sentence.
- Probability is a way of **summarizing the uncertainty** regarding a situation.
- The exact probability assigned to a sentence depends on existing **evidence**: the knowledge available up to date.
- Probabilities can change when **more evidence** is acquired.

Probability

Ontological commitments: Facts hold or do not hold in the world.

Epistemological commitments: A probabilistic agent has a **degree of belief** in a particular sentence. Degrees of belief range from 0 (for sentences that are certainly false) to 1 (for sentences that are certainly true).

Basic Axioms of Probability Theory

Let A and B be two events such that $A \cap B /= \emptyset$. Then:

$$P(A \cup B) = P(A) + P(B)$$

Example - Question

Let us consider the experiment of throwing two fair dice. What is the probability that the total of the two numbers that will appear on the dice is 11?

Example - Answer

$$P(Total = 11) = P((5,6)) + P((6,5)) = 1/36 + 1/36 = 2/36 = 1/18$$

Unconditional vs. Conditional Probability

- **Unconditional** or **prior** probabilities refer to degrees of belief in propositions in the absence of any other information.

  Example: $P(Total = 11)$

- **Conditional** or **posterior** probabilities refer to degrees of belief in propositions given some more information which is usually called **evidence**.

  Example: $P(Doubles = true \mid Die_1 = 5)$

Conditional Probability

- Conditional probabilities are defined in terms of unconditional ones.
- For any propositions a and b such that $P(b) > 0$, we define the **(conditional) probability of a given b** as follows:

$$P(a|b) = \frac{P(a \land b)}{P(b)}$$

- From the above definition we have:

$$P(a \land b) = P(a|b)\,P(b)$$

This equation is traditionally called the **product rule**.

Example

$$P(Doubles = true \mid Dice_1 = 5) = \frac{P(Doubles = true \wedge Dice_1 = 5)}{P(Dice_1 = 5)} =$$

$$\frac{1/36}{1/6} = 1/6$$

Random Variables

- **Random variables** are variables that take values assigned to elements of a sample space (outcomes of an experiment in the traditional probability jargon).
- A random variable takes its values from a certain **domain**.

  Examples:

  - The domain of $Dice_1$ is $\{1, \ldots, 6\}$.
  - The domain of $Weather = \{sunny, rainy, cloudy, snowy\}$.
  - A Boolean random variable has the domain $\{true, false\}$.
- **Notation:** The names of random variables (e.g., $Total$) will start with an upper case letter.

Random Variables (cont'd)

- Random variables can be **discrete** (with finite or countably infinite domain) or **continuous** (the domain is a subset of $\mathbb{R}$).

  We will only consider discrete random variables.

Probability Distribution of a Random Variable

- A **probability distribution** or **probability density function (p.d.f.)** of a random variable X is a function that tells us how the probability mass (i.e., total mass of 1) is allocated across the values that X can take.
- For a discrete random variable X, a probability density function is the function $f(x) = P(X = x)$.
- All the values of a probability density function for X are given by the vector $\mathbf{P}(X)$.

Example

$$P(Weather = sunny) = 0.6$$

$$P(Weather = rainy) = 0.1$$

$$P(Weather = cloudy) = 0.29$$

$$P(Weather = snowy) = 0.01$$

Equivalently as a vector:

$$\mathbf{P}(Weather) = \langle 0.6, 0.1, 0.29, 0.01 \rangle$$

Example (cont'd)

Equivalently as a table:

	$P(\cdot)$
$Weather = sunny$	0.6
$Weather = rainny$	0.1
$Weather = cloudy$	0.29
$Weather = snowy$	0.01

The Joint Probability Distribution

If we have more than one random variable and we are considering problems that involve two or more of these variables at the same time, then the **joint probability distribution** specifies degrees of belief in the values that these functions take **jointly**.

The joint probability distribution $\mathbf{P}(\mathbf{X})$, where $\mathbf{X}$ is a vector of random variables, is usually specified graphically by a n-dimensional table (where n is the dimension of $\mathbf{X}$).

Example: (two Boolean variables *Toothache* and *Cavity*)

	toothache	$\neg$*toothache*
cavity	0.04	0.06
$\neg$*cavity*	0.01	0.89

Independence

The notion of independence captures the situation when the probability of a random variable taking a certain value is **not influenced** by the fact that we know the value of some other variable.

Definition. Two propositions a and b are called **independent** if $P(a|b) = P(a)$ (equivalently: $P(b|a) = P(b)$ or $P(a \wedge b) = P(a)P(b)$).

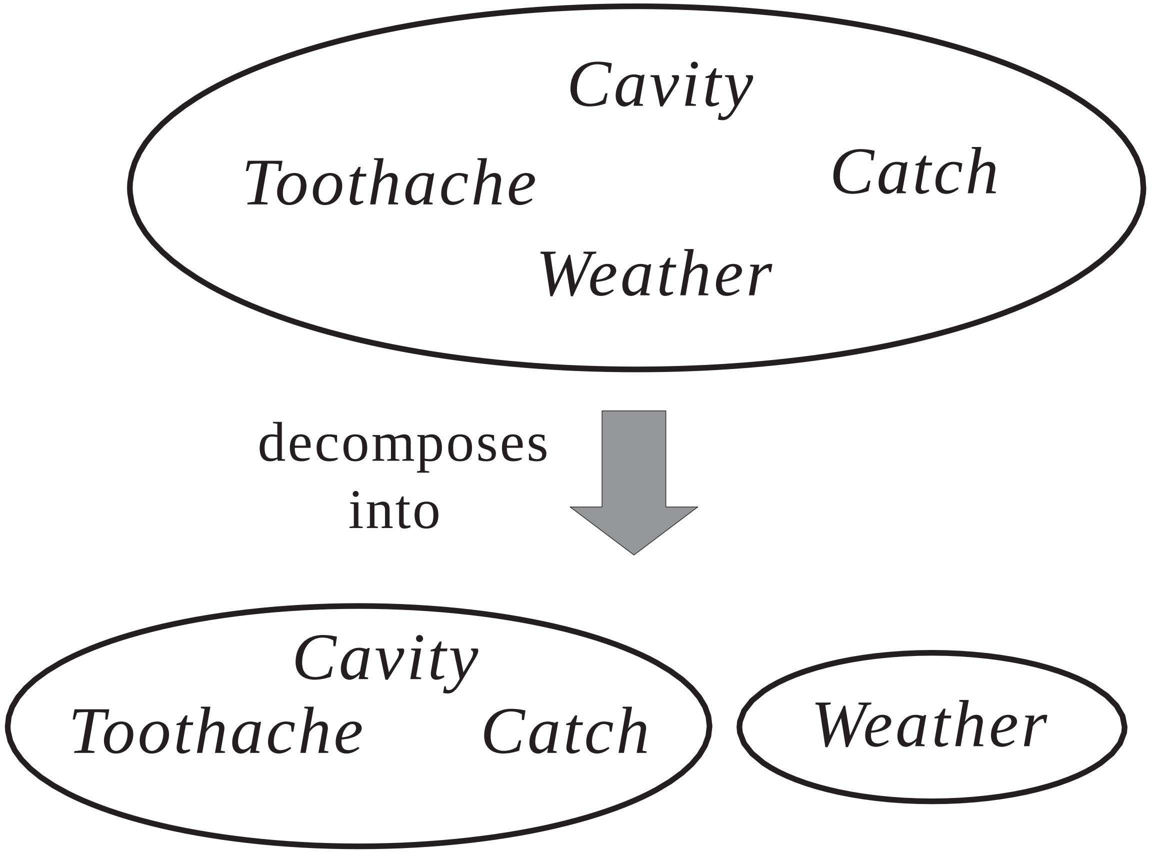
Definition. Two random variables X and Y are called **independent** if $\mathbf{P}(X \mid Y) = \mathbf{P}(X)$ (equivalently: $\mathbf{P}(Y \mid X) = \mathbf{P}(Y)$ or $\mathbf{P}(X, Y) = \mathbf{P}(X)\,\mathbf{P}(Y)$).

Example

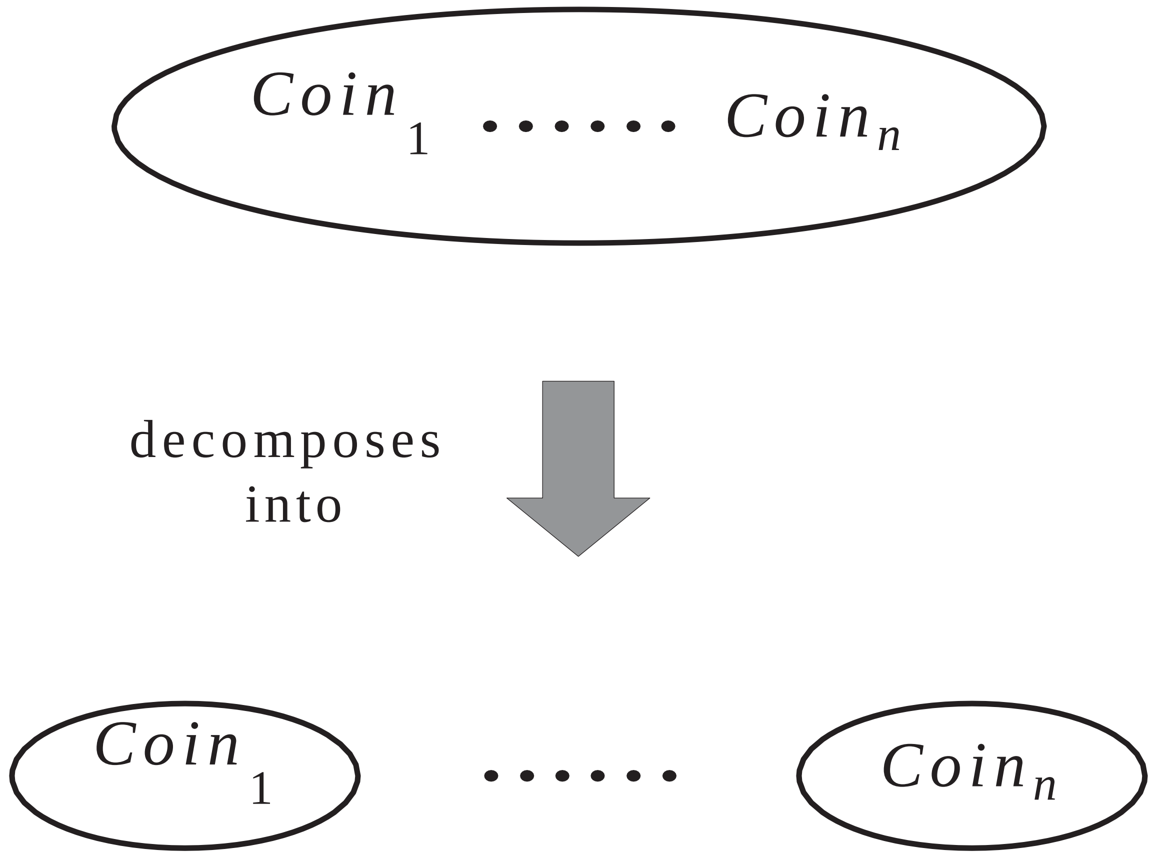
$$\mathbf{P}(Weather \mid Toothache, Catch, Cavity) = \mathbf{P}(Weather)$$

Note: Zeus might be an exception to this rule!

Examples


Cavity
Toothache
Catch
Weather
decomposes into
Cavity
Toothache
Catch
Weather

Examples (cont'd)



Bayes' Rule

From the product rule, we have:

$$P(a \wedge b) = P(a|b)P(b) \quad \text{and} \quad P(a \wedge b) = P(b|a)P(a)$$

If we equate the right hand sides and divide by $P(a)$, we have **Bayes' rule**:

$$P(b|a) = \frac{P(a|b)P(b)}{P(a)}$$

Bayes' rule is the **basis of most modern probabilistic inference systems.**

Applying Bayes' Rule: The Simple Case

In many applications, we perceive as evidence the effect of some unknown cause and we would like to determine that cause. In this case, Bayes' rule can help:

$$P(cause \mid effect) = \frac{P(effect \mid cause)\, P(cause)}{P(effect)}$$

Comments:

- $P(effect \mid cause)$ quantifies the relationship between $cause$ and $effect$ in a **causal** way.
- $P(cause \mid effect)$ does the same thing in a **diagnostic** way.

Application: Medical Diagnosis

A doctor might have the following knowledge:

- Meningitis causes a stiff neck in 70% of the patients.
- The probability that a patient has meningitis is 1/50000.
- The probability that a patient has a stiff neck is 1/100.

Then, we can use Bayes' rule to **compute the probability that a patient has meningitis given that he has a stiff neck.**

Medical Diagnosis (cont'd)

$$P(m \mid s) = \frac{P(s \mid m)\ P(m)}{P(s)} = \frac{0.7 * 1/50000}{0.01} = 0.0014$$

The probability is very small.

There has been much work in using Bayesian networks for medical diagnosis.

Conditional Independence

Observation: *Toothache* and *Catch* **are independent given the presence or absence of a cavity.**

Explanation: The presence or absence of a cavity directly causes toothache or the catching of the steel probe. However, **the two variables do not directly depend on each other** if we take *Cavity* into account. The existence of toothache depends on the state of the nerves of the teeth, while, whether the steel probe catches in a tooth, depends on the skill of the dentist.

Conditional Independence (cont'd)

Formally:

$$\mathbf{P}(toothache \wedge catch \,|\, Cavity) =$$

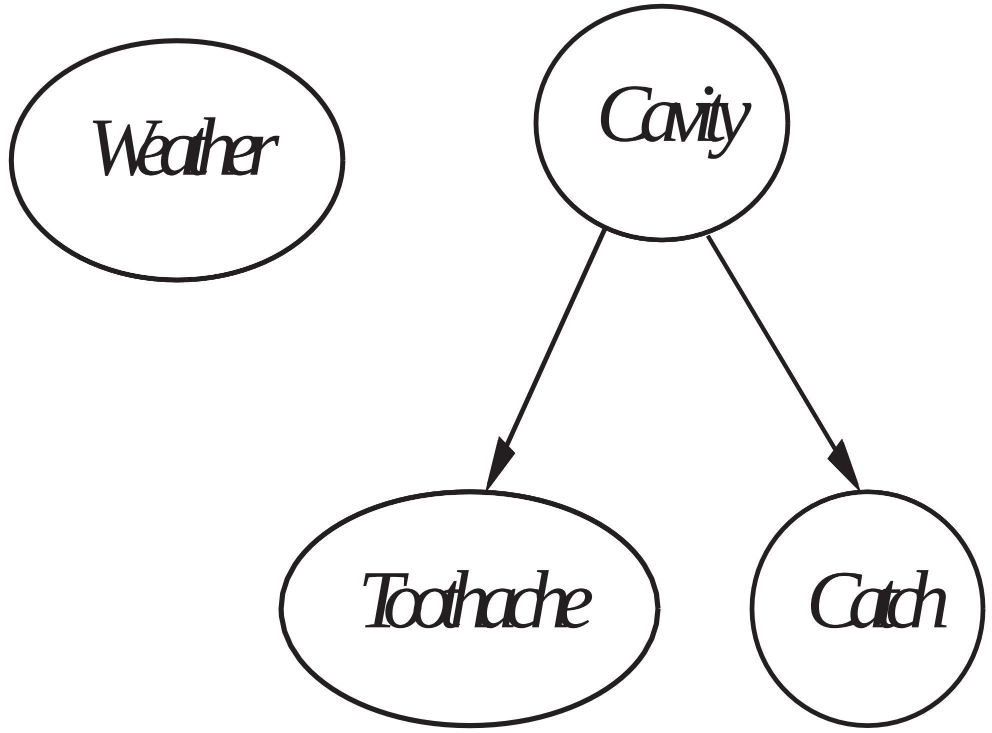
$$\mathbf{P}(toothache \,|\, Cavity)\ \mathbf{P}(catch \,|\, Cavity)$$

This property is called the **conditional independence** of *toothache* and *catch* given *Cavity*.

Bayesian Networks

- Bayesian networks are graphical representations that allow us to represent explicitly **causal, independence** and **conditional independence** relationships and **reason with them efficiently**.
- Bayesian networks enable us to represent **qualitative** (causality, independence) and **quantitative** (probabilistic) **knowledge**.
- Other terms for the same thing: belief networks, probabilistic networks, causal networks etc.

Example



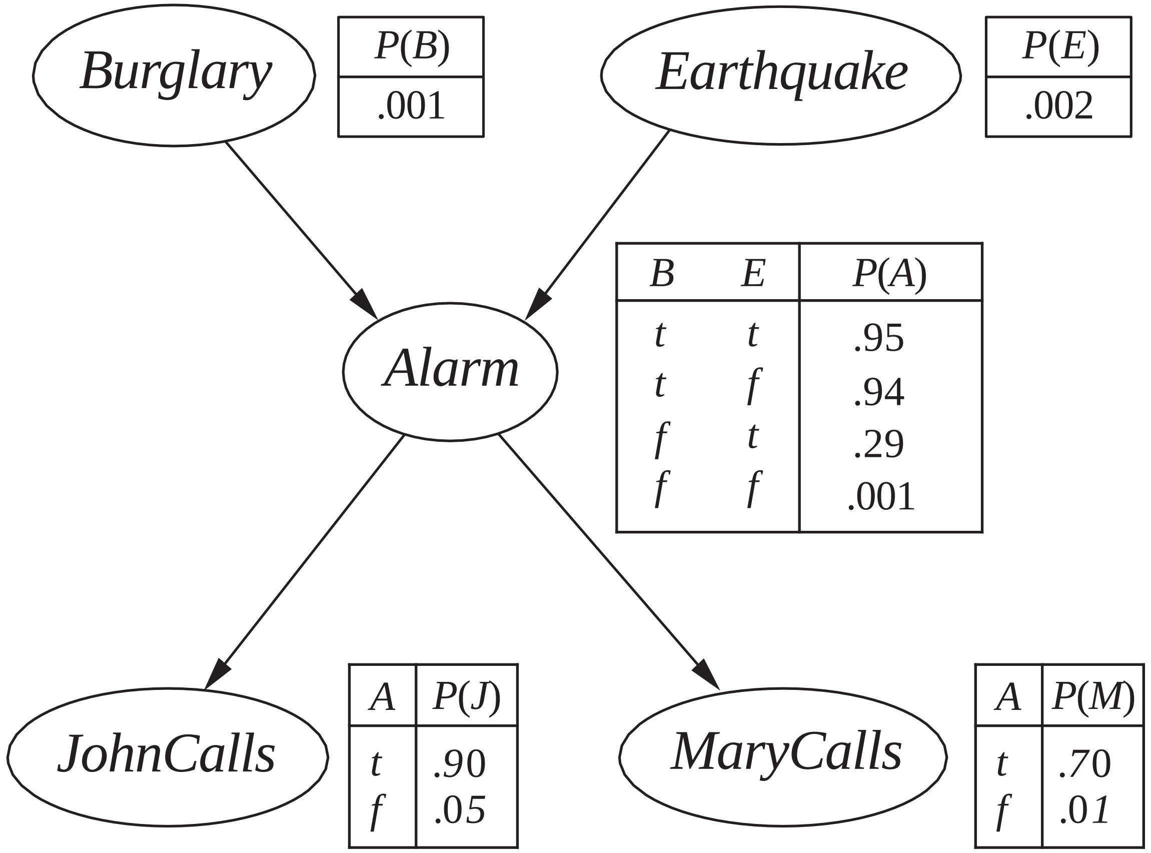
Bayesian Networks

Definition. A **Bayesian network** is a directed acyclic graph $G = (V, E)$ where:

- Each node $X \in V$ is a random variable (discrete or continuous).
- Each directed edge $(X, Y) \in E$ indicates a **causal dependency** between variables X and Y (i.e., X **directly influences** Y).
- Each node Y has a **conditional probability distribution** $P(Y \mid Parents(Y))$ that quantifies this dependency.

Terminology: If there is an edge $(X, Y) \in E$, X is called the **parent** of Y and Y is called the **child** of X. $Parents(X)$ will be used to denote the parents of a node X.

Example



$P(B)$
.001

$P(E)$
.002

B	E	$P(A)$
t	t	.95
t	f	.94
f	t	.29
f	f	.001

A	$P(J)$
t	.90
f	.05

A	$P(M)$
t	.70
f	.01

How do we Construct Bayesian Networks

In related theory and practice, the following methods have been used:

- Using expert knowledge
- Automatic synthesis from some other formal representation of domain knowledge (e.g., in reliability analysis or diagnosis of hardware systems).
- Learning from data (e.g., in medicine).

Applications of Bayesian Networks

- Medicine
- Engineering
- Networking
- Bioinformatics

Representing Uncertainty

Chapter 13

Probability Theory

- Probability theory serves as a formal means for
  - Representing and reasoning with uncertain knowledge
  - Modeling **degrees of belief** in a proposition (event, conclusion, diagnosis, etc.)

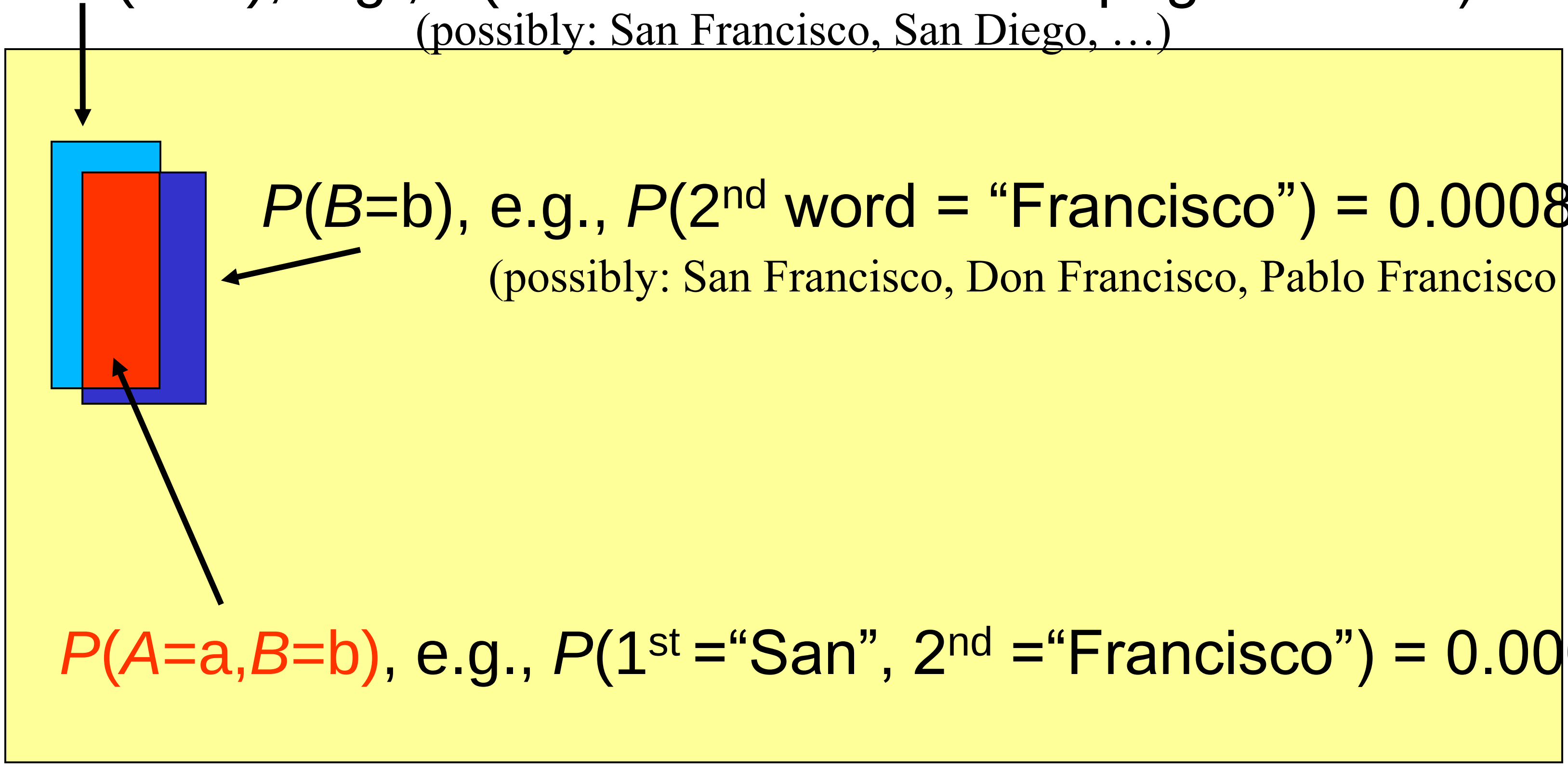
- *Probability is the "language" of uncertainty*
  - A key modeling method in modern AI

Joint Probability

- The **joint probability** *P*(*A*=a, *B*=b) is shorthand for *P*(*A*=a ∧ *B*=b), i.e., the probability of *both* *A*=a and *B*=b happening

P(*A*=a), e.g., *P*(1st word on a random page = “San”) = 0.001

(possibly: San Francisco, San Diego, …)



P(*B*=b), e.g., *P*(2nd word = “Francisco”) = 0.0008

(possibly: San Francisco, Don Francisco, Pablo Francisco …)

P(*A*=a,*B*=b), e.g., *P*(1st =“San”, 2nd =“Francisco”) = 0.0007

Full Joint Probability Distribution

Weather

Temp		*sunny*	*cloudy*	*rainy*
	hot	150/365	40/365	5/365
	cold	50/365	60/365	60/365

- *P*(*Temp=hot*, *Weather=rainy*) = *P*(*hot*, *rainy*) = 5/365 = 0.014
- The **full joint probability distribution** table for *n* random variables, each taking *k* values, has k^n entries

Full Joint Probability Distribution

Bird	*Flier*	*Young*	**Probability**
T	T	T	0.0
T	T	F	0.2
T	F	T	0.04
T	F	F	0.01
F	T	T	0.01
F	T	F	0.01
F	F	T	0.23
F	F	F	0.5

Sums to 1

3 Boolean random variables $\Rightarrow 2^3 - 1 = 7$ “degrees of freedom” (DOF) or “independent values”

Computing Conditional Probability

$P(\neg B|F) \quad = \ ?$

$P(F) \quad = \ ?$

Note: $P(\neg B|F)$ means $P(B\text{=false} \mid F\text{=true})$ and $P(F)$ means $P(F\text{=true})$

Computing Conditional Probability

$P(\neg B|F) = P(\neg B, F)/P(F)$

$= (P(\neg B, F, Y) + P(\neg B, F, \neg Y))/P(F)$

$= (0.01 + 0.01)/P(F)$

$P(F) = P(F, B, Y) + P(F, B, \neg Y) + P(F, \neg B, Y) + P(F, \neg B, \neg Y)$

$= 0.0 + 0.2 + 0.01 + 0.01$

$= 0.22$

Marginalization

Bayes’s Rule

- Bayes’s Rule is the basis for probabilistic reasoning given a prior model of the world, P(Q), and a new piece of evidence, E, Bayes’s rule says how this piece of evidence decreases our ignorance about the world
- Initially, know P(Q)   (“prior”)
- Update after knowing E (“posterior”):

$$P(Q|E) = P(Q)\frac{P(E|Q)}{P(E)}$$

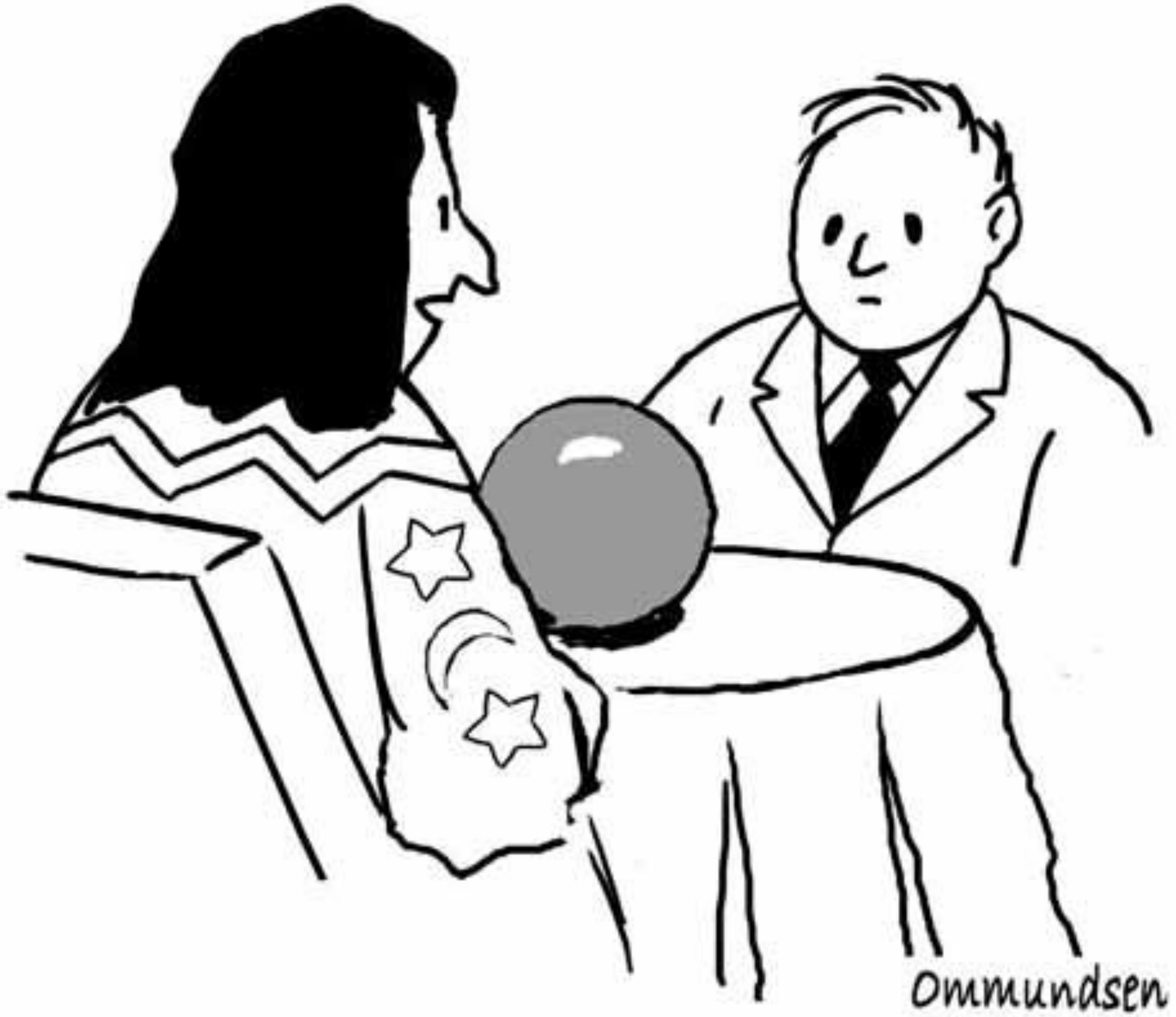
Inference with Bayes's Rule

***P*(*A*|*B*) = *P*(*B* | *A*)*P*(*A*) / *P*(*B*)** **Bayes's rule**

- Why do we make things this complicated?
  - Often *P*(*B*|*A*), *P*(*A*), *P*(*B*) are easier to get
  - Some names:
    - **Prior *P*(*A*)**: probability of *A* *before* any evidence
    - **Likelihood *P*(*B*|*A*)**: assuming *A*, how likely is the evidence
    - **Posterior *P*(*A*|*B*)**: probability of *A* after knowing evidence *B*
    - **(Deductive) Inference**: deriving an unknown probability from known ones
- If we have the full joint probability table, we can simply compute *P*(*A*|*B*) = *P*(*A*, *B*) / *P*(*B*)

Bayes’s Rule in Practice

$P_{d_1}(s_1\;\bar{s}_5 \ldots s_j) = \dfrac{P_{d_1}\cdot P_{s_1|d_1}(1-P_{s_5|d_1})\ldots P_{s_j|d_1}}{\sum_{\text{all } k} P_{d_k} P_{s_1|d_k}(1-P_{s_5|d_k})\ldots P_{s_j|d_k}}$
R. Morgan



"Is this needed for a Bayesian analysis?"

Summary of Important Rules

- **Conditional Probability**: $P(A|B) = P(A,B)/P(B)$
- **Product rule**: $P(A,B) = P(A|B)P(B)$
- **Chain rule**: $P(A,B,C,D) = P(A|B,C,D)P(B|C,D)P(C|D)P(D)$
- **Conditionalized version of Chain rule**:

  $$P(A,B|C) = P(A|B,C)P(B|C)$$
- **Bayes's rule**: $P(A|B) = P(B|A)P(A)/P(B)$
- **Conditionalized version of Bayes's rule**:

  $$P(A|B,C) = P(B|A,C)P(A|C)/P(B|C)$$
- **Addition / Conditioning rule**: $P(A) = P(A,B) + P(A,\neg B)$

  $$P(A) = P(A|B)P(B) + P(A|\neg B)P(\neg B)$$

Common Mistake

- $P(A) = 0.3$ so $P(\neg A) = 1 - P(A) = 0.7$

- $P(A|B) = 0.4$ so $P(\neg A|B) = 1 - P(A|B) = 0.6$

  because $P(A|B) + P(\neg A|B) = 1$

  but $P(A|\neg B) \neq 0.6$ (in general)

  because $P(A|B) + P(A|\neg B) \neq 1$ in general

Independence

Two events A, B are **independent** if the following hold:

- $P(A, B) = P(A) * P(B)$
- $P(A, \neg B) = P(A) * P(\neg B)$

  …
- $P(A \mid B) = P(A)$
- $P(B \mid A) = P(B)$
- $P(A \mid \neg B) = P(A)$

  …

Independence

- Independence is a kind of domain knowledge
  - Needs an understanding of ***causation***
  - Very strong assumption

- Example: *P*(burglary) = 0.001 and *P*(earthquake) = 0.002
  - Let's say they are *independent*
  - The full joint probability table = ?

Independence

- Given: $P(B) = 0.001$, $P(E) = 0.002$, $P(B|E) = P(B)$
- The full joint probability distribution table (FJPD) is:

Burglary	Earthquake	Prob.
B	E	
B	$\neg E$	
$\neg B$	E	
$\neg B$	$\neg E$	

- Need only 2 numbers to fill in entire table
- Now we can do anything, since we have the FJPD

Conditional Independence

- Random variables can be *dependent*, but **conditionally independent**
- Example: Your house has an alarm
  - Neighbor John will call when he hears the alarm
  - Neighbor Mary will call when she hears the alarm
  - Assume John and Mary don’t talk to each other
- Is *JohnCall independent* of *MaryCall*?
  - **No** – If John called, it is likely the alarm went off, which increases the probability of Mary calling
  - *P*(*MaryCall* | *JohnCall*) ¯ *P*(*MaryCall*)

Conditional Independence

- But, if we *know* the status of the *alarm*, *JohnCall* will ***not*** affect whether or not Mary calls

  $P(MaryCall \mid Alarm, JohnCall) = P(MaryCall \mid Alarm)$

- We say *JohnCall* and *MaryCall* are **conditionally independent** given *Alarm*
- In general, "*A* and *B* are conditionally independent given *C*" means:

  $P(A \mid B, C) = P(A \mid C)$

  $P(B \mid A, C) = P(B \mid C)$

  $P(A, B \mid C) = P(A \mid C)\, P(B \mid C)$

Independence vs. Conditional Independence

- Say Alice and Bob each toss ***separate coins***. *A* represents "Alice's coin toss is heads" and *B* represents "Bob's coin toss is heads"
- *A* and *B* are **independent**
- Now suppose Alice and Bob toss the ***same coin***. Are *A* and *B independent*?
  - *No*. Say the coin may be biased towards heads. If *A* is heads, it will lead us to increase our belief in *B* beings heads. That is, $P(B|A) > P(A)$

- Say we add a new variable, *C*: "the coin is biased towards heads"
- The values of *A* and *B* are *dependent on C*
- But if we know *for certain* the value of *C* (true or false), then any evidence about *A* cannot change our belief about *B*
- That is, $P(B|C) = P(B|A, C)$
- *A* and *B* are **conditionally independent** given *C*

Naïve Bayes Classifier

- Say we have one class/diagnosis/decision variable, A
- Goal is to find the value of A that is most likely given evidence B, C, D, … :

$$argmax_a\, P(\mathrm{A{=}a})P(B|\mathrm{A{=}a})P(C|\mathrm{A{=}a})P(D|\mathrm{A{=}a})/P(B,C,D)$$

But *P*(B,C,D) is a constant here for all *a*, so instead compute:

$$argmax_a\, P(\mathrm{A{=}a})P(B|\mathrm{A{=}a})P(C|\mathrm{A{=}a})P(D|\mathrm{A{=}a})$$

Naïve Bayes Classifier

- Find *v* =
  $\text{argmax}_v P(Y = v) \prod_{i=1}^{n} P(X_i = u_i | Y = v)$

  Class variable

  Evidence variable

- Assumes all evidence variables are conditionally independent of each other given the class variable
- Robust since it gives the right answer as long as the correct class is more likely than all others

Naïve Bayes Classifier

- Assume k classes and n evidence variables, each with r possible values
- k-1 values needed for computing $P(Y=v)$
- rk values needed for computing $P(X_i=u_i \mid Y=v)$ for each evidence variable X_i
- So, (k-1) + nrk values needed instead of exponential size FJPD table

Naïve Bayes Classifier

- Conditional probabilities can be very, very small, so instead use logarithms to avoid underflow:

$$\text{argmax}_v \log P(Y = v) + \sum_{i=1}^{n} \log P(X_i = u_i | Y = v)$$

Summary of Important Rules

- **Conditional Probability**: $P(A|B) = P(A,B)/P(B)$
- **Product rule**: $P(A,B) = P(A|B)P(B)$
- **Chain rule**: $P(A,B,C,D) = P(A|B,C,D)P(B|C,D)P(C|D)P(D)$
- **Conditionalized version of Chain rule**:

$$P(A,B|C) = P(A|B,C)P(B|C)$$

- **Bayes's rule**: $P(A|B) = P(B|A)P(A)/P(B)$
- **Conditionalized version of Bayes's rule**:

$$P(A|B,C) = P(B|A,C)P(A|C)/P(B|C)$$

- **Addition / Conditioning rule**: $P(A) = P(A,B) + P(A, \neg B)$

$$P(A) = P(A|B)P(B) + P(A|\neg B)P(\neg B)$$

Bayesian Networks

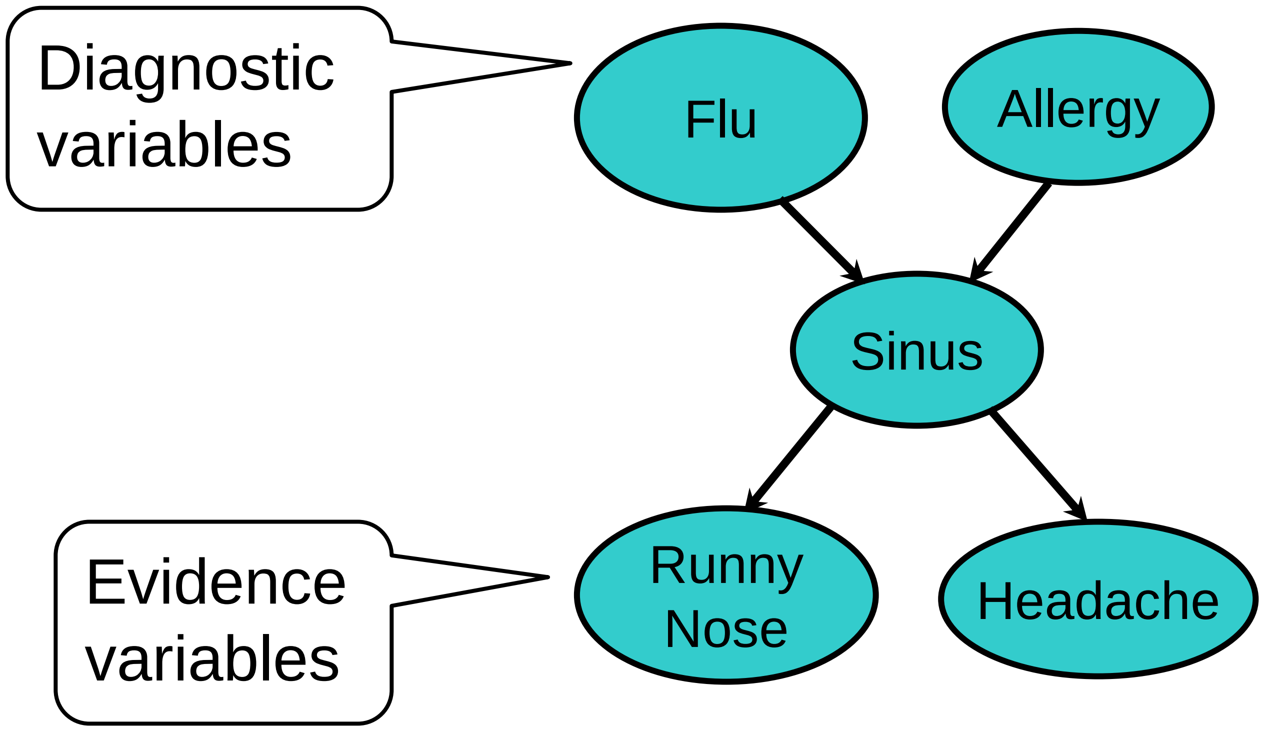
Chapter 14.1, 14.2, and 14.4
”

[based on slides by Jerry Zhu and Andrew Moore]

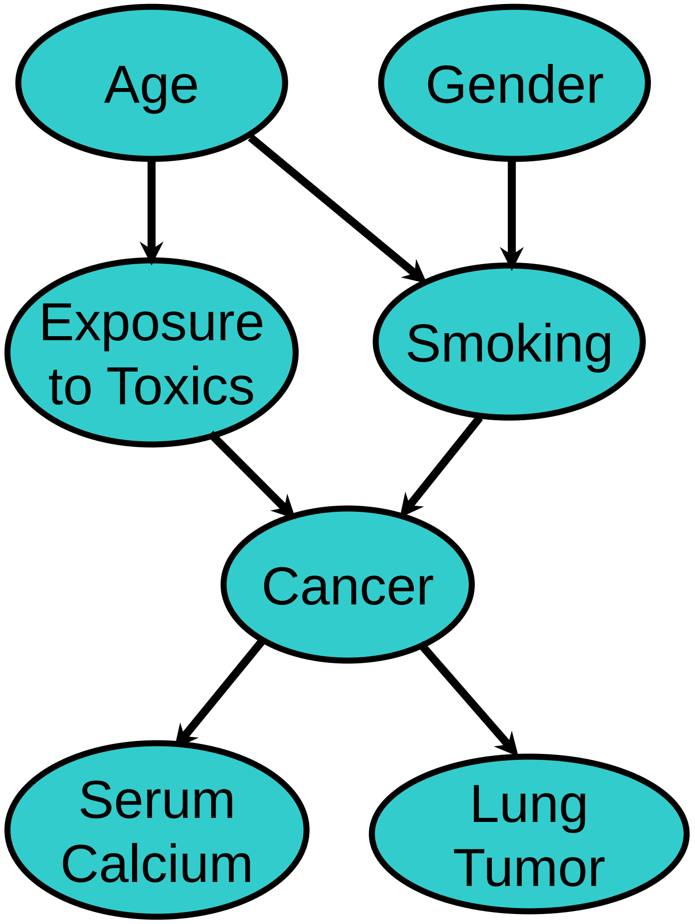
Introduction

- Probabilistic models allow us to use probabilistic inference (e.g., Bayes's rule) to compute the probability distribution over a set of unobserved ("hypothesis") given a set of observed variables
- Full joint probability distribution table is great for inference in an uncertain world, but is terrible to obtain and store
- Bayesian Networks allow us to represent joint distributions in manageable chunks using
  - Independence, conditional independence
- Bayesian Network can do any inference

A Bayesian Network



A Bayesian Network

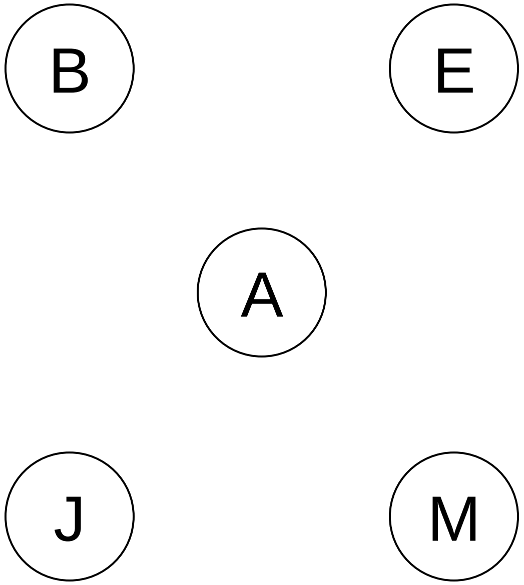


Applications

- Medical diagnosis systems
- Manufacturing system diagnosis
- Computer systems diagnosis
- Network systems diagnosis
- Helpdesk troubleshooting
- Information retrieval
- Customer modeling

Creating a Bayes Net

- **Step 1**: Add variables. Choose the variables you want to include in the Bayes Net



B: there's burglary in your house

E: there's an earthquake

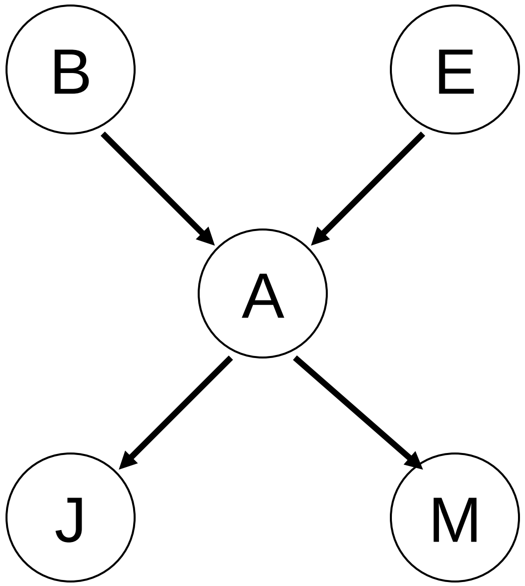
A: your alarm sounds

J: your neighbor John calls you

M: your other neighbor Mary calls you

Creating a Bayes Net

- **Step 2**: Add directed edges
  - The graph must be **acyclic**
  - If node *X* is given parents $Q_1, \ldots, Q_m$, you are promising that any variable that's ***not*** a *descendant* of *X* is conditionally independent of *X* given $Q_1, \ldots, Q_m$



B: there's burglary in your house

E: there's an earthquake

A: your alarm sounds

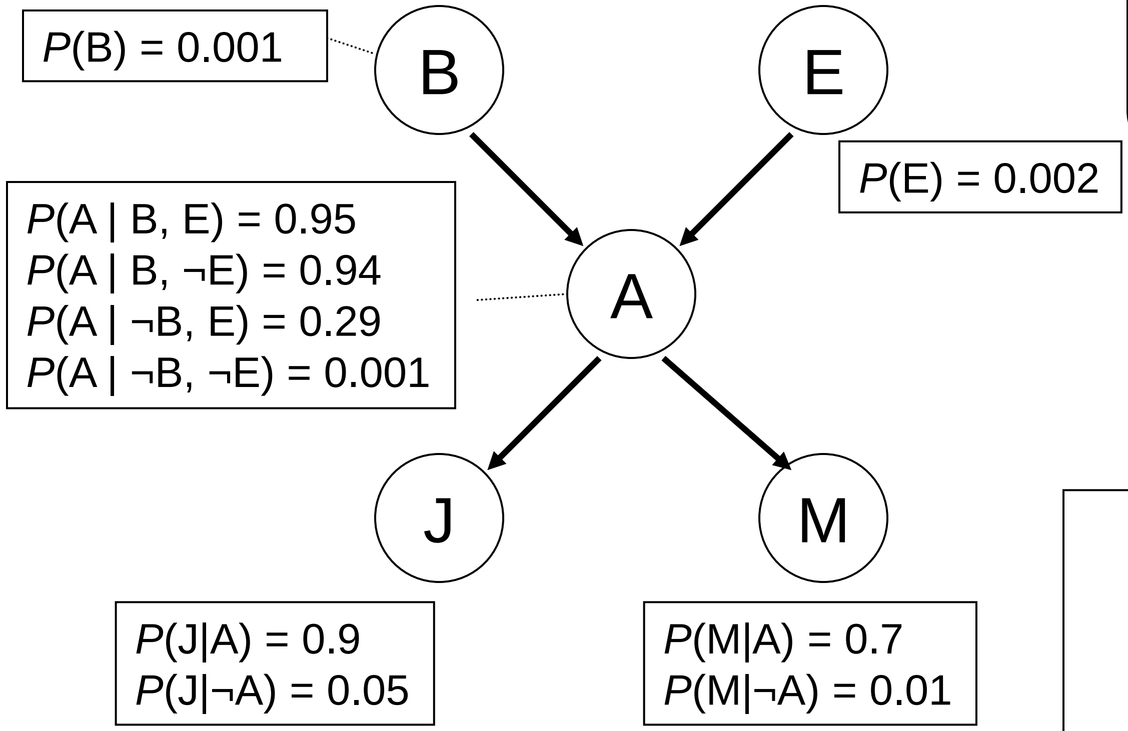
J: your neighbor John calls you

M: your other neighbor Mary calls you

Creating a Bayes Net

- **Step 3**: Add CPT's
- Each table must list *P*(*X* | Parent values) for all combinations of parent values

e.g., you must specify *P*(J|A) AND *P*(J|¬A) since they don't have to sum to 1!



B: there's burglary in your house

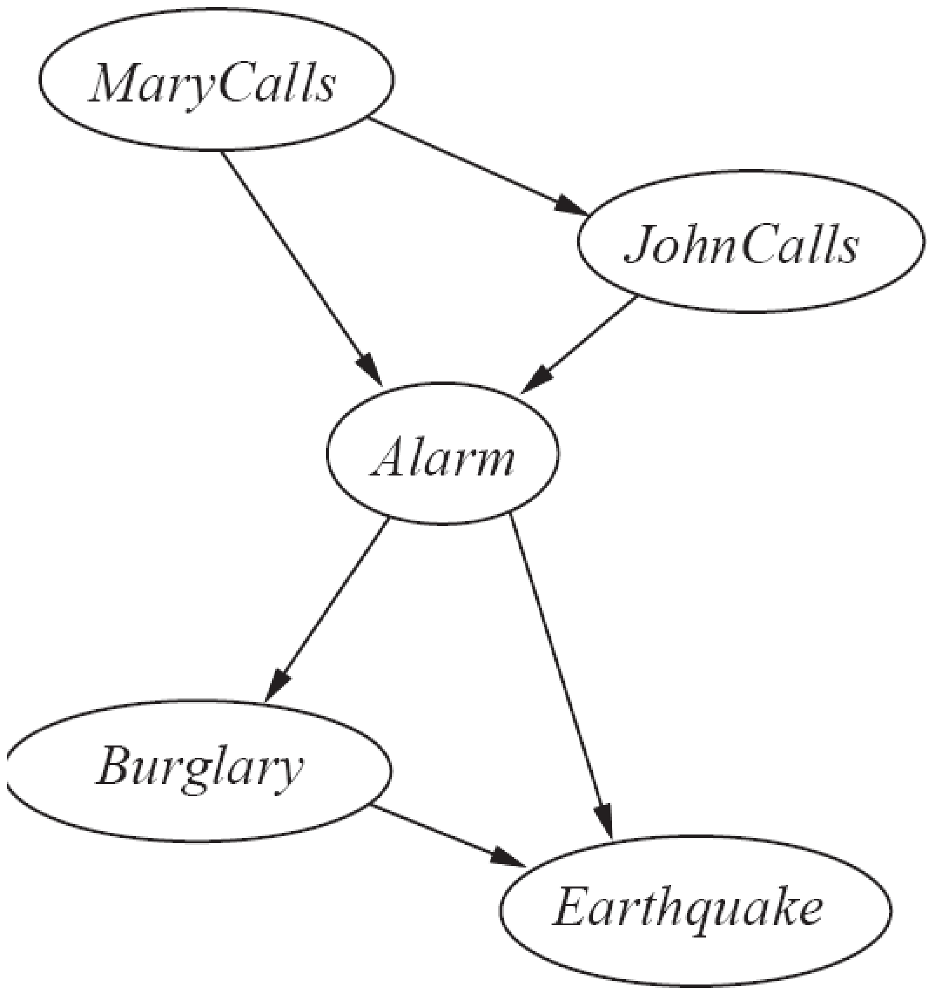
E: there's an earthquake

A: your alarm sounds

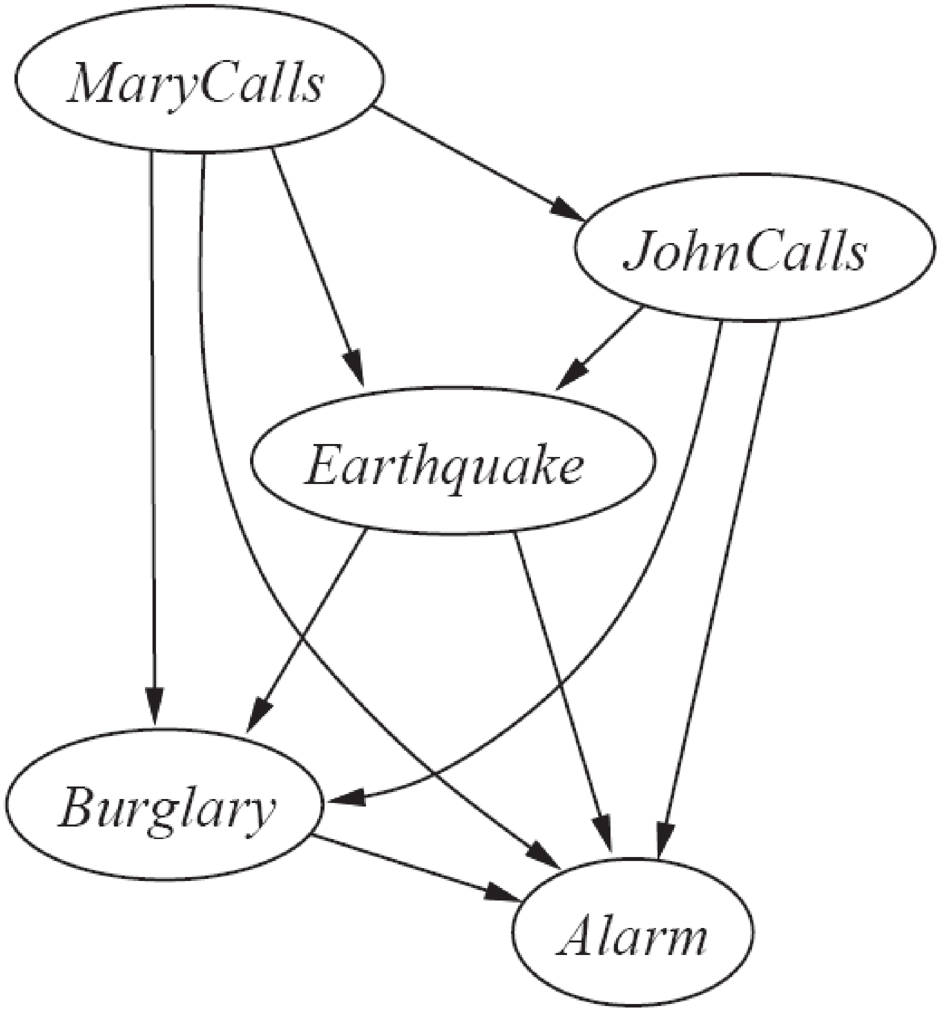
J: your neighbor John calls you

M: your other neighbor Mary calls you

The Bayesian Network Created from a Different Variable Ordering

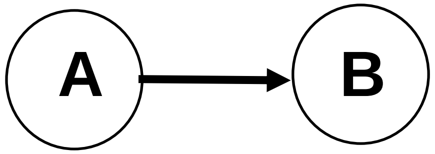


The Bayesian Network Created from a Different Variable Ordering



Variable Dependencies

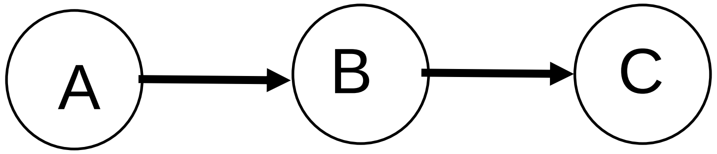
- Directed arc from one variable to another variable



- Is A guaranteed to be *independent* of B?
  - No – Information can be transmitted over 1 arc
    - Example: My knowing the Alarm went off, increases my belief there has been a Burglary, and similarly, my knowing there has been a Burglary increases my belief the Alarm went off

Causal Chain

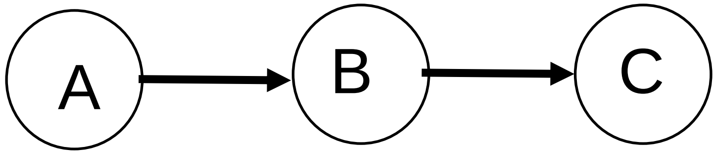
- This local configuration is called a “**causal chain**:”



- Is A guaranteed to be *independent* of C?
  - No – Information can be transmitted between A and C through B if B is *not* observed
    - Example: Not knowing Alarm means that my knowing that a Burglary has occurred increases my belief that Mary calls, and similarly, knowing that Mary Calls increases my belief that there has been a Burglary

Causal Chain

- This local configuration is called a "**causal chain**:"

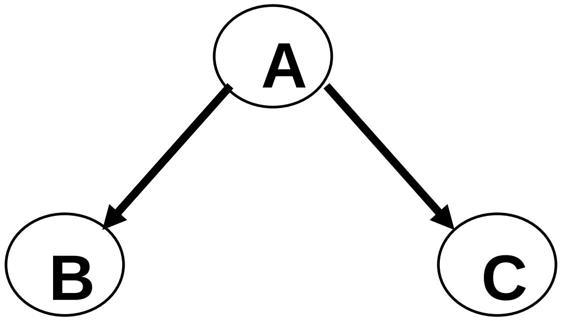


$$\begin{tikzcd} A \arrow[r] & B \arrow[r] & C \end{tikzcd}$$

- Is A *independent* of C *given B*?
  - Yes – Once B is observed, information *cannot* be transmitted between A and C through B; B "blocks" the information path; "C is conditionally independent of A given B"
    - Example: Knowing that the Alarm went off means that also knowing that a Burglary has taken place will **not** increase my belief that Mary Calls

Common Cause

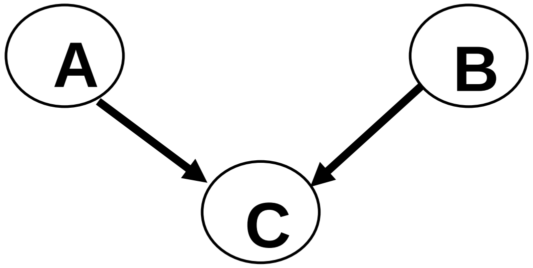
- This configuration is called "**common cause**:"



- Is it guaranteed that B and C are *independent*?
  - No – Information can be transmitted through A to the children of A if A is *not* observed
- Is it guaranteed that B and C are independent *given A*?
  - Yes – Observing the cause, A, blocks the influence between effects B and C; "B is conditionally independent of C given A"

Common Effect

- This configuration is called "**common effect**:"

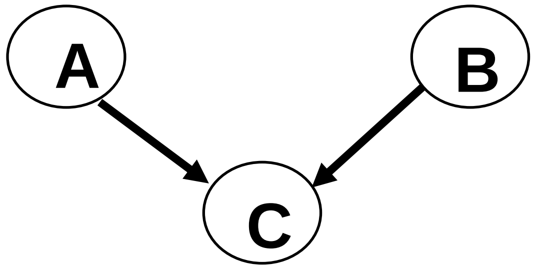


- Are A and B *independent*?
  - Yes
    - Example: Burglary and Earthquake cause the Alarm to go off, but they are not correlated
  - Proof: $P(a,b) = \Sigma_c P(a,b,c)$ by marginalization

    $= \Sigma_c P(a) P(b|a) P(c|a,b)$ by chain rule

Common Effect

- This configuration is called "**common effect**:"



- Are A and B independent *given C*?
  - No – Information can be transmitted through C among the parents of C if C *is* observed
    - Example: If I already know that the Alarm went off, my further knowing that there has been an Earthquake, *decreases* my belief that there has been a Burglary. Called "explaining away."
  - Similarly, if C has descendant D and D is given, then A and B are *not* independent

Neural networks are parallel computing devices, which is basically an attempt to make a computer model of the brain. The main objective is to develop a system to perform various computational tasks faster than the traditional systems. These tasks include pattern recognition and classification, approximation, optimization, and data clustering.

What is Artificial Neural Network?

Artificial Neural Network ANN

is an efficient computing system whose central theme is borrowed from the analogy of biological neural networks. ANNs are also named as “artificial neural systems,” or “parallel distributed processing systems,” or “connectionist systems.” ANN acquires a large collection of units that are interconnected in some pattern to allow communication between the units. These units, also referred to as nodes or neurons, are simple processors which operate in parallel.

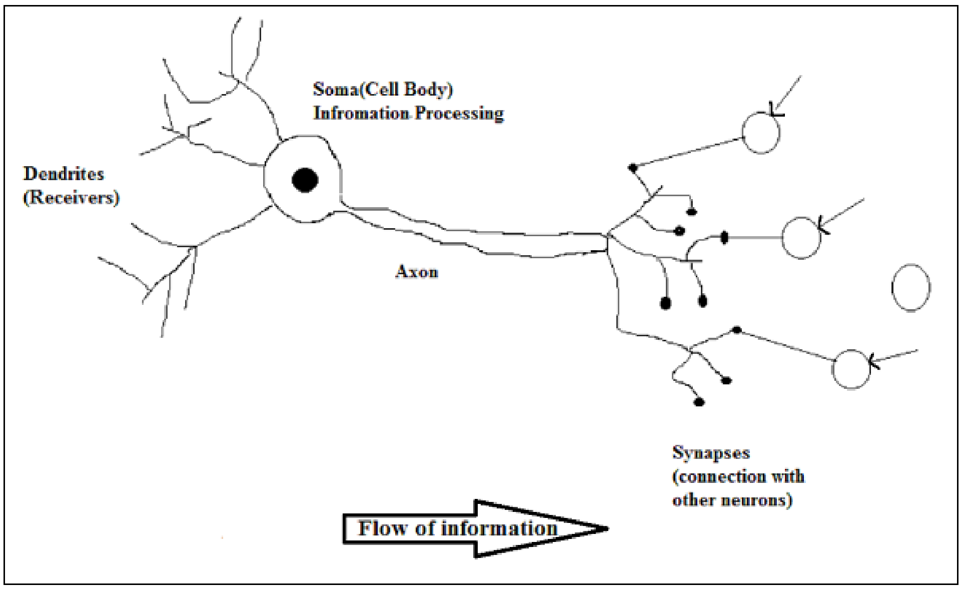
Every neuron is connected with other neuron through a connection link. Each connection link is associated with a weight that has information about the input signal. This is the most useful information for neurons to solve a particular problem because the weight usually excites or inhibits the signal that is being communicated. Each neuron has an internal state, which is called an activation signal. Output signals, which are produced after combining the input signals and activation rule, may be sent to other units.

Biological Neuron

A nerve cell $neuron$

is a special biological cell that processes information. According to an estimation, there are huge number of neurons, approximately 10^{11} with numerous interconnections, approximately 10^{15}.

Schematic Diagram



Working of a Biological Neuron

As shown in the above diagram, a typical neuron consists of the following four parts with the help of which we can explain its working −

- **Dendrites** − They are tree-like branches, responsible for receiving the information from other neurons it is connected to. In other sense, we can say that they are like the ears of neuron.
- **Soma** − It is the cell body of the neuron and is responsible for processing of information, they have received from dendrites.
- **Axon** − It is just like a cable through which neurons send the information.
- **Synapses** − It is the connection between the axon and other neuron dendrites.

ANN versus BNN

Before taking a look at the differences between Artificial Neural Network ANN

and Biological Neural Network BNN

, let us take a look at the similarities based on the terminology between these two.

Biological Neural Network BNN	**Artificial Neural Network** ANN
Soma	Node
Dendrites	Input
Synapse	Weights or Interconnections
Axon	Output

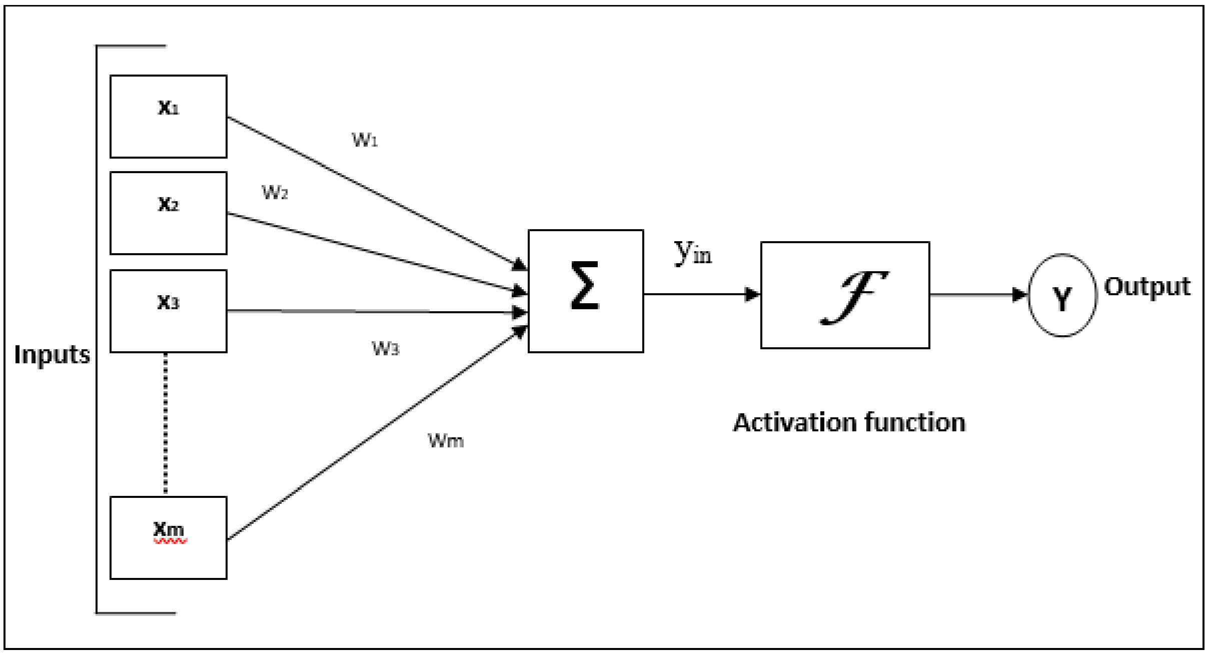
The following table shows the comparison between ANN and BNN based on some criteria mentioned.

Criteria	**BNN**	**ANN**
Processing	Massively parallel, slow but superior than ANN	Massively parallel, fast but inferior than BNN
Size	10^{11} neurons and 10^{15} interconnections	10^2 to 10^4 nodes *mainly depends on the type of application and network designer.*

Learning	They can tolerate ambiguity	Very precise, structured and formatted data is required to tolerate ambiguity
Fault tolerance	Performance degrades with even partial damage	It is capable of robust performance, hence has the potential to be fault tolerant
Storage capacity	Stores the information in the synapse	Stores the information in continuous memory locations

Model of Artificial Neural Network

The following diagram represents the general model of ANN followed by its processing.



For the above general model of artificial neural network, the net input can be calculated as follows −

$y_{in} = x_1.w_1 + x_2.w_2 + x_3.w_3 \ldots x_m.w_m$

i.e., Net input $y_{in} = \sum x_i.w_i$

The output can be calculated by applying the activation function over the net input.

$Y = F(y_{in})$

Output = function *net input calculated*

Processing of ANN depends upon the following three building blocks −

- Network Topology
- Adjustments of Weights or Learning
- Activation Functions

In this chapter, we will discuss in detail about these three building blocks of ANN

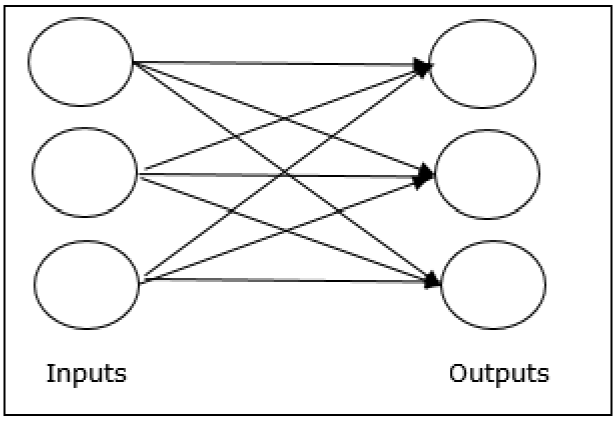
Network Topology

A network topology is the arrangement of a network along with its nodes and connecting lines. According to the topology, ANN can be classified as the following kinds −

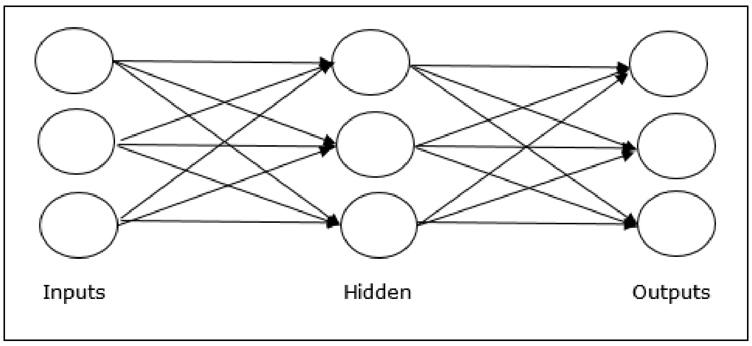
Feedforward Network

It is a non-recurrent network having processing units/nodes in layers and all the nodes in a layer are connected with the nodes of the previous layers. The connection has different weights upon them. There is no feedback loop means the signal can only flow in one direction, from input to output. It may be divided into the following two types −

- **Single layer feedforward network** − The concept is of feedforward ANN having only one weighted layer. In other words, we can say the input layer is fully connected to the output layer.



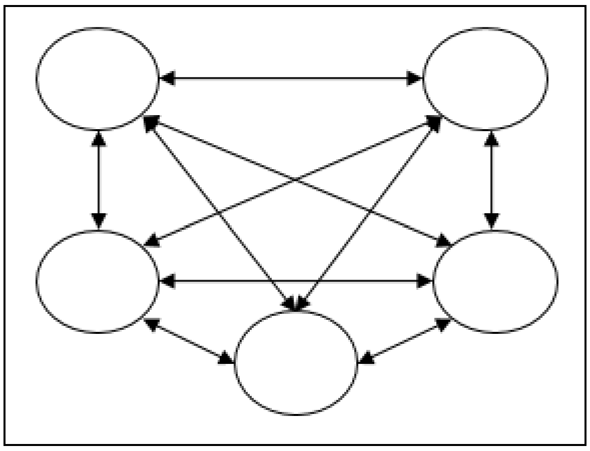
- **Multilayer feedforward network** − The concept is of feedforward ANN having more than one weighted layer. As this network has one or more layers between the input and the output layer, it is called hidden layers.



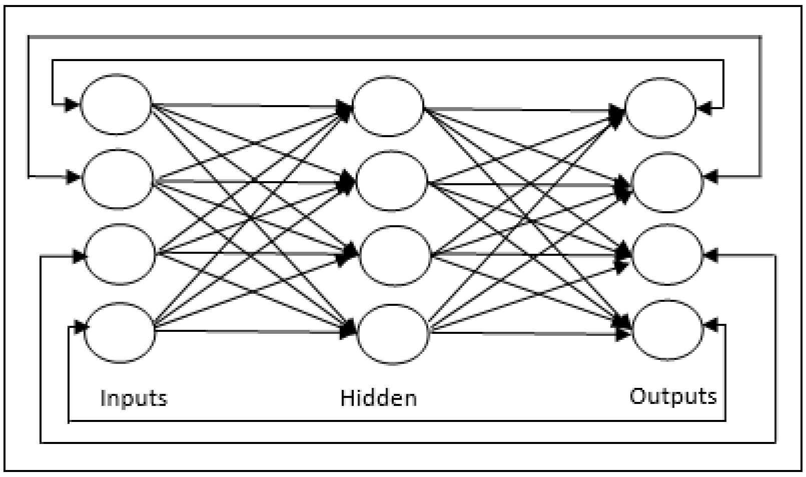
Feedback Network

As the name suggests, a feedback network has feedback paths, which means the signal can flow in both directions using loops. This makes it a non-linear dynamic system, which changes continuously until it reaches a state of equilibrium. It may be divided into the following types –

- **Recurrent networks** − They are feedback networks with closed loops. Following are the two types of recurrent networks.
- **Fully recurrent network** − It is the simplest neural network architecture because all nodes are connected to all other nodes and each node works as both input and output.



- **Jordan network** − It is a closed loop network in which the output will go to the input again as feedback as shown in the following diagram.



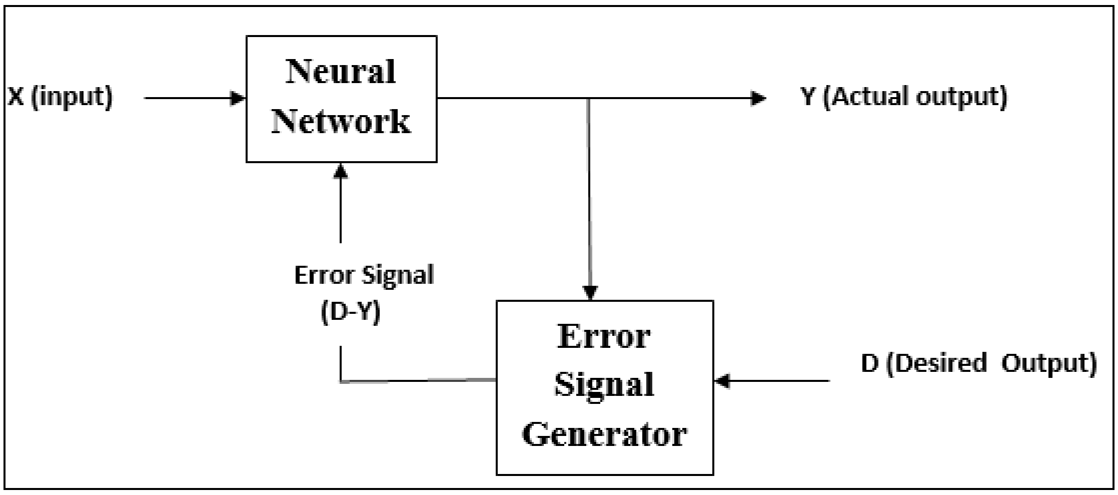
Adjustments of Weights or Learning

Learning, in artificial neural network, is the method of modifying the weights of connections between the neurons of a specified network. Learning in ANN can be classified into three categories namely supervised learning, unsupervised learning, and reinforcement learning.

Supervised Learning

As the name suggests, this type of learning is done under the supervision of a teacher. This learning process is dependent.

During the training of ANN under supervised learning, the input vector is presented to the network, which will give an output vector. This output vector is compared with the desired output vector. An error signal is generated, if there is a difference between the actual output and the desired output vector. On the basis of this error signal, the weights are adjusted until the actual output is matched with the desired output.

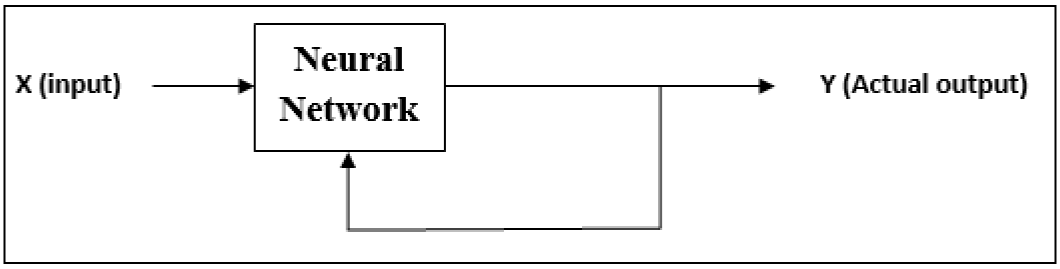


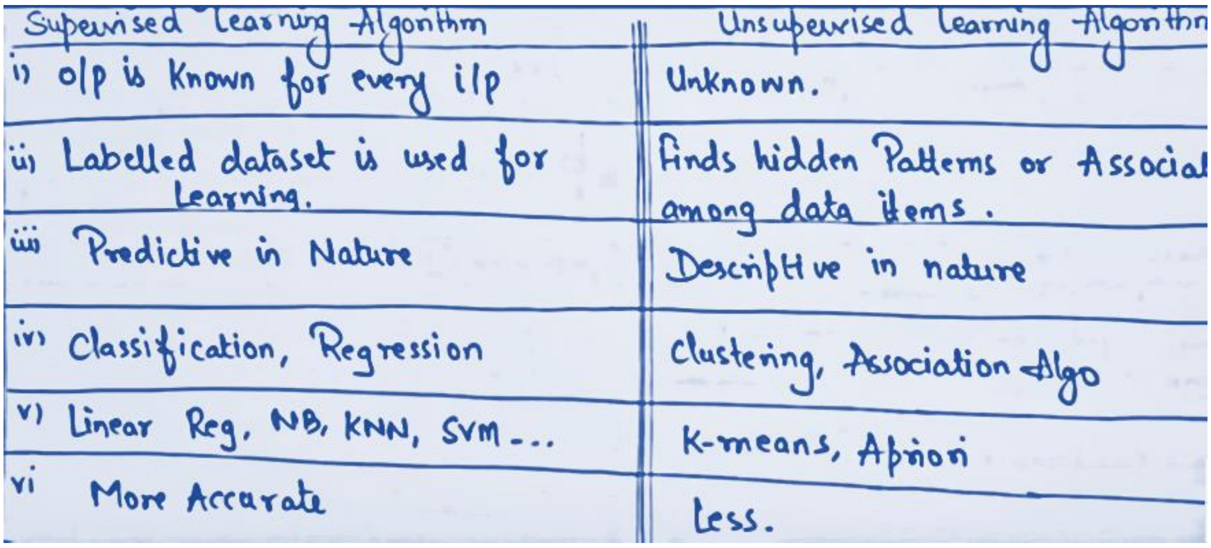
Unsupervised Learning

As the name suggests, this type of learning is done without the supervision of a teacher. This learning process is independent.

During the training of ANN under unsupervised learning, the input vectors of similar type are combined to form clusters. When a new input pattern is applied, then the neural network gives an output response indicating the class to which the input pattern belongs.

There is no feedback from the environment as to what should be the desired output and if it is correct or incorrect. Hence, in this type of learning, the network itself must discover the patterns and features from the input data, and the relation for the input data over the output.



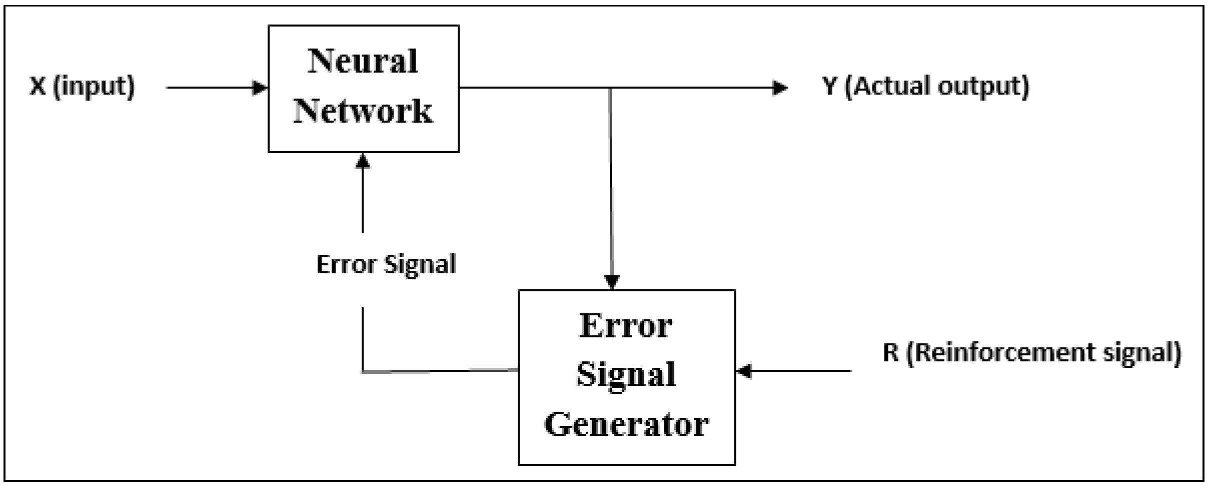


Supervised Learning Algorithm	Unsupervised Learning Algorith
i) o/p is Known for every i/p	Unknown.
ii) Labelled dataset is used for Learning.	Finds hidden Patterns or Associa among data items.
iii) Predictive in Nature	Descriptive in nature
iv) Classification, Regression	Clustering, Association Algo
v) Linear Reg, NB, KNN, SVM ...	K-means, Apriori
vi More Accurate	Less.

Reinforcement Learning

As the name suggests, this type of learning is used to reinforce or strengthen the network over some critic information. This learning process is similar to supervised learning, however we might have very less information.

During the training of network under reinforcement learning, the network receives some feedback from the environment. This makes it somewhat similar to supervised learning. However, the feedback obtained here is evaluative not instructive, which means there is no teacher as in supervised learning. After receiving the feedback, the network performs adjustments of the weights to get better critic information in future.



Activation Functions

It may be defined as the extra force or effort applied over the input to obtain an exact output. In ANN, we can also apply activation functions over the input to get the exact output. Followings are some activation functions of interest −

Linear Activation Function

It is also called the identity function as it performs no input editing. It can be defined as −

$F(x)=x$

Sigmoid Activation Function

It is of two type as follows −

- **Binary sigmoidal function** − This activation function performs input editing between 0 and 1. It is positive in nature. It is always bounded, which means its output cannot be less than 0 and more than 1. It is also strictly increasing in nature, which means more the input higher would be the output. It can be defined as

$$F(x) = sigm(x) = \frac{1}{1 + exp(-x)}$$

- **Bipolar sigmoidal function** − This activation function performs input editing between -1 and 1. It can be positive or negative in nature. It is always bounded, which means its output cannot be less than -1 and more than 1. It is also strictly increasing in nature like sigmoid function. It can be defined as

$$F(x) = sigm(x) = \frac{2}{1 + exp(-x)} - 1 = \frac{1 - exp(x)}{1 + exp(x)}$$

Branches in AI

Major areas of AI

- Fuzzy Logic
- Natural Language Processing (NLP)
- Expert System
- Robotics

Natural Language Processing (NLP)

- Natural Language Processing (NLP) refers to AI method of communicating with an intelligent systems using a natural language such as English.

- **Components of NLP**

There are two components of NLP as given

1.Natural Language Understanding (NLU)
2.Natural Language Generation (NLG)

Components of NLP

- **Natural Language Understanding (NLU)**

Understanding involves the following tasks −

1. Mapping the given input in natural language into useful representations.
2. Analysing different aspects of the language.

- **Natural Language Generation (NLG)**

It is the process of producing meaningful phrases and sentences in the form of natural language from some internal representation.

Expert System

- The expert systems are the computer applications developed to solve complex problems in a particular domain, at the level of extra-ordinary human intelligence and expertise.

- **Characteristics of Expert Systems**

➢ High performance
➢ Understandable
➢ Reliable
➢ Highly responsive

Capabilities of Expert System

The expert systems are capable of

- Advising
- Instructing and assisting human in decision making
- Demonstrating
- Diagnosing
- Predicting results
- Justifying the conclusion

Components of Expert System

The components of ES include

- **Knowledge Base -** It contains domain-specific and high-quality knowledge.
- **Inference Engine -** It acquires and manipulates the knowledge from the knowledge base to arrive at a particular solution.
- **User Interface -** User interface provides interaction between user of the ES and the ES itself with the help of natural language processing.

Robotics

Robotics is a branch of AI, which is composed of Electrical Engineering, Mechanical Engineering, and Computer Science for designing, construction, and application of robots.

Aspects of Robotics

- The robots have **mechanical construction**, form, or shape designed to accomplish a particular task.
- They have **electrical components** which power and control the machinery.
- They contain some level of **computer program** that determines what, when and how a robot does something.

Applications of Robotics

- Industry
- Military
- Medicine
- Exploration
- Entertainment

Fuzzy Logic

Outline

- Fundamental fuzzy concepts
- Fuzzy propositional and predicate logic
- Fuzzification
- Defuzzification
- Fuzzy control systems
- Types of fuzzy algorithms
- Applications of fuzzy logic

Introduction

- Fuzzy concepts first introduced by Zadeh in the 1960s and 70s
- Traditional computational logic and set theory is all about
  - true or false
  - zero or one
  - in or out (in terms of set membership)
  - black or white (no grey)
- Not the case with fuzzy logic and fuzzy sets!

Basic Concepts

- Approximation ("granulation")
  - A colour can be described precisely using RGB values, or it can be approximately described as "red", "blue", etc.
- Degree ("graduation")
  - Two different colours may both be described as "red", but one is considered to be more red than the other
- Fuzzy logic attempts to reflect the human way of thinking

Terminology

- Fuzzy set
  - A set X in which each element y has a grade of membership $\mu_X(y)$ in the range 0 to 1, i.e. set membership may be partial
    e.g. if *cold* is a fuzzy set, exact temperature values might be mapped to the fuzzy set as follows:
    - 15 degrees $\rightarrow$ 0.2 (slightly cold)
    - 10 degrees $\rightarrow$ 0.5 (quite cold)
    - 0 degrees $\rightarrow$ 1 (totally cold)
- Fuzzy relation
  - Relationships can also be expressed on a scale of 0 to 1
    e.g. degree of *resemblance* between two people

Terminology (cont'd)

- Fuzzy variable
  - Variable with (labels of) fuzzy sets as its values
- Linguistic variable
  - Fuzzy variable with values that are words or sentences in a language
    e.g. variable *colour* with values *red*, *blue*, *yellow*, *green*…
- Linguistic hedge
  - Term used as a modifier for basic terms in linguistic values
    e.g. words such as *very*, *a bit*, *rather*, *somewhat*, etc.

Formal Fuzzy Logic

- Fuzzy logic can be seen as an extension of ordinary logic, where the main difference is that we use fuzzy sets for the membership of a variable
- We can have fuzzy propositional logic and fuzzy predicate logic
- Fuzzy logic can have many advantages over ordinary logic in areas like artificial intelligence where a simple true/false statement is insufficient

Traditional Logic

- Propositional logic:
  - Propositional logic is a formal system that uses true statements to form or prove other true statements
  - There are two types of sentences: simple sentences and compound sentences
  - Simple sentences are propositional constants; statements that are either true or false
  - Compound sentences are formed from simpler sentences by using negations ¬, conjunctions ∧, disjunctions ∨, implications ⇒, reductions ⇐, and equivalences ⇔
- Predicate logic:
  - Onto propositional logic, this adds the ability to quantify variables, so we can manipulate statements about all or some things
  - Two common quantifiers are the existential ∃ and universal ∀ quantifiers

Formal Fuzzy Logic

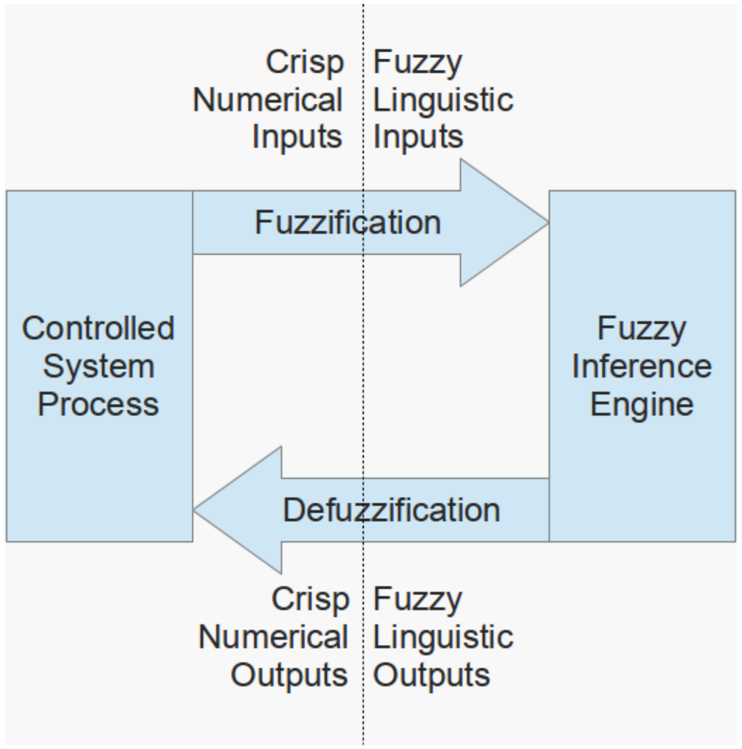
- Fuzzy Propositional Logic
  - Like ordinary propositional logic, we introduce propositional variables, truth-functional connectives, and a propositional constant 0
  - Some of these include:
    - Monoidal t-norm-based propositional fuzzy logic
    - Basic propositional fuzzy logic
    - Łukasiewicz fuzzy logic
    - Gödel fuzzy logic
    - Product fuzzy logic
    - Rational Pavelka logic
- Fuzzy Predicate Logic
  - These extend fuzzy propositional logic by adding universal and existential quantifiers in a manner similar to the way that predicate logic is created from propositional logic

Simple Fuzzy Operators

- As described by Zadeh (1973)…
- NOT X = 1 - $\mu_X(y)$
  - e.g. 0.8 cold → (1 – 0.8) = 0.2 NOT cold
- X ORY (union) = max($\mu_X(y)$, $\mu_Y(y)$)
  - e.g. 0.8 cold, 0.5 rainy → 0.8 cold OR rainy
- X ANDY (intersection) = min($\mu_X(y)$,$\mu_Y(y)$)
  - e.g. 0.9 hot, 0.7 humid → 0.7 hotAND humid

Fuzzy System Overview

- When making inferences, we want to clump the continuous numerical values into sets
- Unlike Boolean logic, fuzzy logic uses fuzzy sets rather than crisp sets to determine the membership of a variable
- This allows values to have a degree of membership with a set, which denotes the extent to which a proposition is true
- The membership function may be triangular, trapezoidal, Gaussian or any other shape

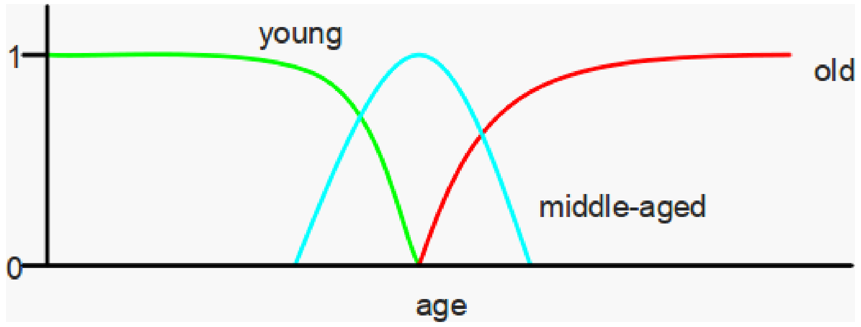


Fuzzification

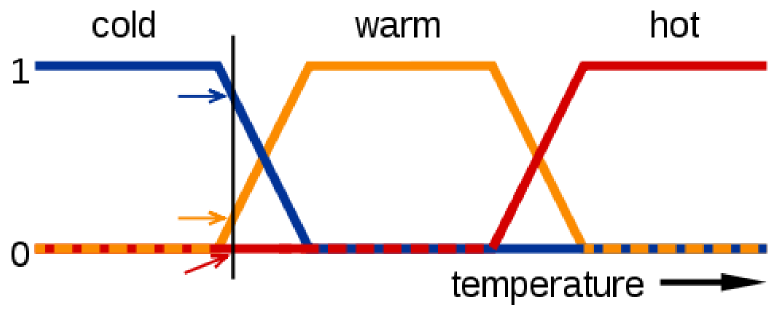
- To apply fuzzy inference, we need our input to be in linguistic values
- These linguistic values are represented by the degree of membership in the fuzzy sets
- The process of translating the measured numerical values into fuzzy linguistic values is called fuzzification
- In other words, fuzzification is where membership functions are applied, and the degree of membership is determined

Membership Functions

- There are largely three types of fuzzifiers:
  - singleton fuzzifier,
  - Gaussian fuzzifier, and
  - trapezoidal or triangular fuzzifier



Gaussian



Trapezoidal

Defuzzification

- Defuzzification is the process of producing a quantifiable result in fuzzy logic
- The fuzzy inference will output a fuzzy result, described in terms of degrees of membership of the fuzzy sets
- Defuzzification interprets the membership degrees in the fuzzy sets into a specific action or real-value

Fuzzy Control Systems

- A simple example application: an automated cooling fan that can adjust its speed according to the room temperature
- Temperature values are mapped to the fuzzy sets *cold*, *warm* and *hot* based on the following mapping functions for each:

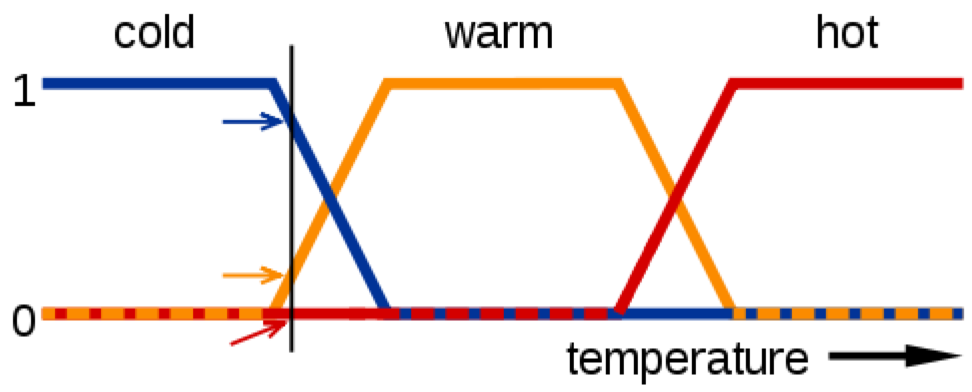


Image: http://en.wikipedia.org/w/index.php?title=File:Fuzzy_logic_temperature_en.svg&page=1

Fuzzy Control Systems

- The inference engine in a fuzzy system consists of linguistic rules
- The linguistic rules consist of two parts:
  - an antecedent block (the conditions), which consists of the linguistic variables
  - a consequent block (the output)

- Fuzzy algorithm
  - Algorithm that includes at least some fuzzy instructions, such as conditional or unconditional action statements
  - Fuzzy conditional statement (A ⟶ B)
    - Conditional statement in which A and/or B are fuzzy sets
      e.g. IF temperature is *hot* THEN fan speed is *high*
    - Defined in terms of a fuzzy relation between the respective "universes of discourse" of A and B (compositional rule of inference)
      e.g. relation between temperature groupings and fan speeds

A Simple Fuzzy Algorithm Example

- The algorithm used by our cooling fan controller might look like this:

```
WHILE fan is switched on
        IF cold THEN stop fan
        IF warm AND NOT cooling down THEN increase fan speed slightly
        IF hot THEN increase fan speed substantially
        etc…
```

Applications of Fuzzy Logic

- Artificial Intelligence
  - Robot motion planning
  - Image segmentation
  - Medical diagnosis systems