

# CS-171 Final Review

- **Propositional Logic**
  - (7.1-7.5)
- **First-Order Logic, Knowledge Representation**
  - (8.1-8.5, 9.1-9.2)
- **Probability & Bayesian Networks**
  - (13, 14.1-14.5)
- **Machine Learning**
  - (18.1-18.4)
- Questions on any topic
- Pre-mid-term material if time and class interest
- Please review your quizzes, mid-term, & old tests
  - At least one question from a prior quiz or old CS-171 test will appear on the Final Exam (and all other tests)

# Review Propositional Logic

## Chapter 7.1-7.5

- Definitions:
  - Syntax, Semantics, Sentences, Propositions, Entails, Follows, Derives, Inference, Sound, Complete, Model, Satisfiable, Valid (or Tautology)
- Syntactic Transformations:
  - E.g.,  $(A \Rightarrow B) \Leftrightarrow (\neg A \vee B)$
- Semantic Transformations:
  - E.g.,  $(KB \models \alpha) \equiv (\models (KB \Rightarrow \alpha))$
- Truth Tables:
  - Negation, Conjunction, Disjunction, Implication, Equivalence (Biconditional)
- Inference:
  - By Model Enumeration (truth tables)
  - By Resolution

# Recap propositional logic: Syntax

- Propositional logic is the simplest logic – illustrates basic ideas
- The proposition symbols  $P_1, P_2$  etc are sentences
  - If  $S$  is a sentence,  $\neg S$  is a sentence (negation)
  - If  $S_1$  and  $S_2$  are sentences,  $S_1 \wedge S_2$  is a sentence (conjunction)
  - If  $S_1$  and  $S_2$  are sentences,  $S_1 \vee S_2$  is a sentence (disjunction)
  - If  $S_1$  and  $S_2$  are sentences,  $S_1 \Rightarrow S_2$  is a sentence (implication)
  - If  $S_1$  and  $S_2$  are sentences,  $S_1 \Leftrightarrow S_2$  is a sentence (biconditional)

# Recap propositional logic:

## Semantics

Each model/world specifies true or false for each proposition symbol

E.g.,

$P_{1,2}$	$P_{2,2}$	$P_{3,1}$
false	true	false

With these symbols, 8 possible models can be enumerated automatically.

Rules for evaluating truth with respect to a model  $m$ :

$\neg S$	is true iff	$S$ is false
$S_1 \wedge S_2$	is true iff	$S_1$ is true <b>and</b> $S_2$ is true
$S_1 \vee S_2$	is true iff	$S_1$ is true <b>or</b> $S_2$ is true
$S_1 \Rightarrow S_2$	is true iff	$S_1$ is false <b>or</b> $S_2$ is true
(i.e.,	is false iff	$S_1$ is true <b>and</b> $S_2$ is false)
$S_1 \Leftrightarrow S_2$	is true iff	$S_1 \Rightarrow S_2$ is true <b>and</b> $S_2 \Rightarrow S_1$ is true

Simple recursive process evaluates an arbitrary sentence, e.g.,

$$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = \text{true} \wedge (\text{true} \vee \text{false}) = \text{true} \wedge \text{true} = \text{true}$$

# Recap propositional logic:

## Truth tables for connectives

$P$	$Q$	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	false	true	false	false	true	true
false	true	true	false	true	true	false
true	false	false	false	true	false	false
true	true	false	true	true	true	true

OR:  $P$  or  $Q$  is true or both are true.  
XOR:  $P$  or  $Q$  is true but not both.

Implication is always true  
when the premises are False!

# Recap propositional logic:

## Logical equivalence and rewrite rules

- To manipulate logical sentences we need some rewrite rules.
- Two sentences are **logically equivalent** iff they are true in same models:  $\alpha \equiv \beta$  iff  $\alpha \models \beta$  and  $\beta \models \alpha$

$$\begin{aligned}(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{de Morgan} \\\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{de Morgan} \\(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge\end{aligned}$$

You need to know these !

# Recap propositional logic:

## Entailment

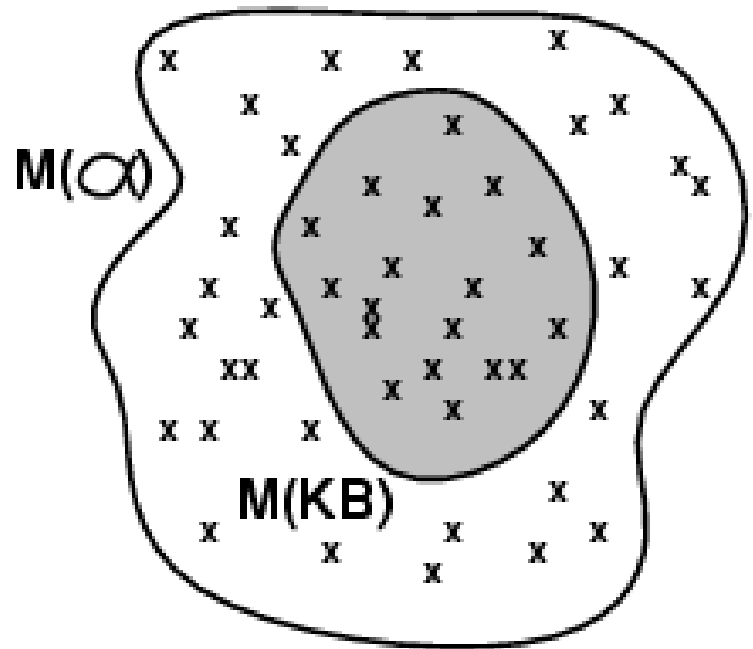
- **Entailment** means that one thing **follows from** another:

$$KB \models \alpha$$

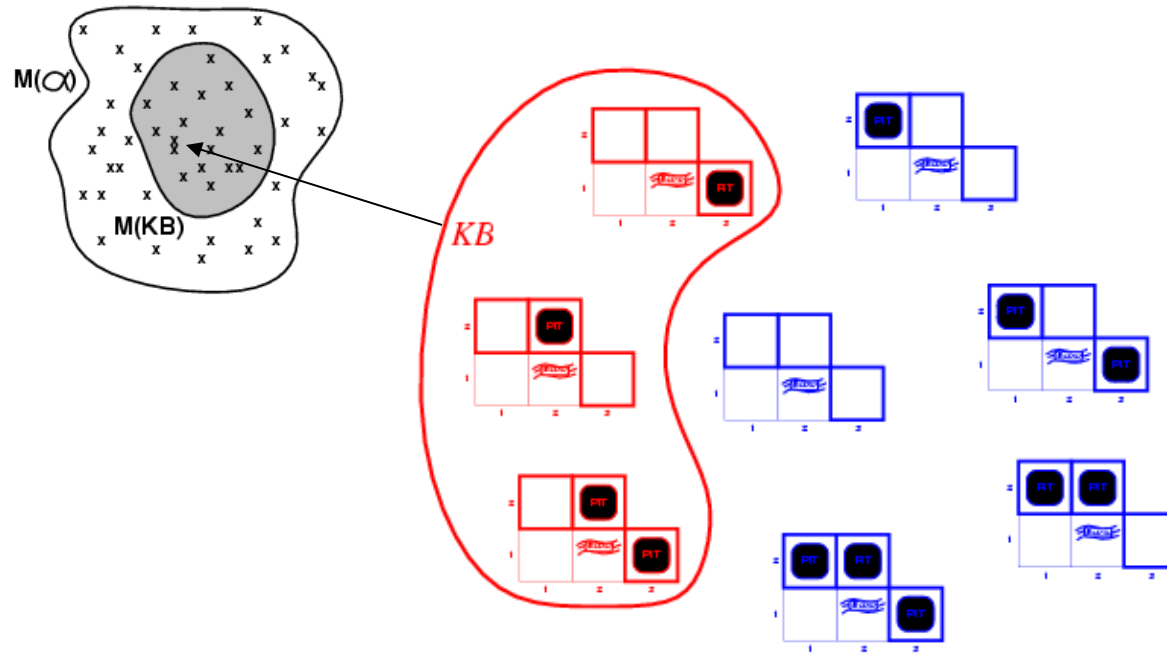
- Knowledge base *KB* entails sentence  $\alpha$  if and only if  $\alpha$  is true in **all worlds** where *KB* is true
  - E.g., the KB containing “the Giants won and the Reds won” entails “The Giants won”.
  - E.g.,  $x+y = 4$  entails  $4 = x+y$
  - E.g., “Mary is Sue’s sister and Amy is Sue’s daughter” entails “Mary is Amy’s aunt.”

# Review: Models (and in FOL, Interpretations)

- **Models** are formal worlds in which truth can be evaluated
- We say  $m$  **is a model of** a sentence  $\alpha$  if  $\alpha$  is true in  $m$
- $M(\alpha)$  is the set of all models of  $\alpha$
- Then  $KB \models \alpha$  iff  $M(KB) \subseteq M(\alpha)$ 
  - E.g.  $KB$ , = “Mary is Sue’s sister and Amy is Sue’s daughter.”
  - $\alpha$  = “Mary is Amy’s aunt.”
- Think of  $KB$  and  $\alpha$  as constraints, and of models  $m$  as possible states.
- $M(KB)$  are the solutions to  $KB$  and  $M(\alpha)$  the solutions to  $\alpha$ .
- Then,  $KB \models \alpha$ , i.e.,  $\models (KB \Rightarrow \alpha)$ , when all solutions to  $KB$  are also solutions to  $\alpha$ .

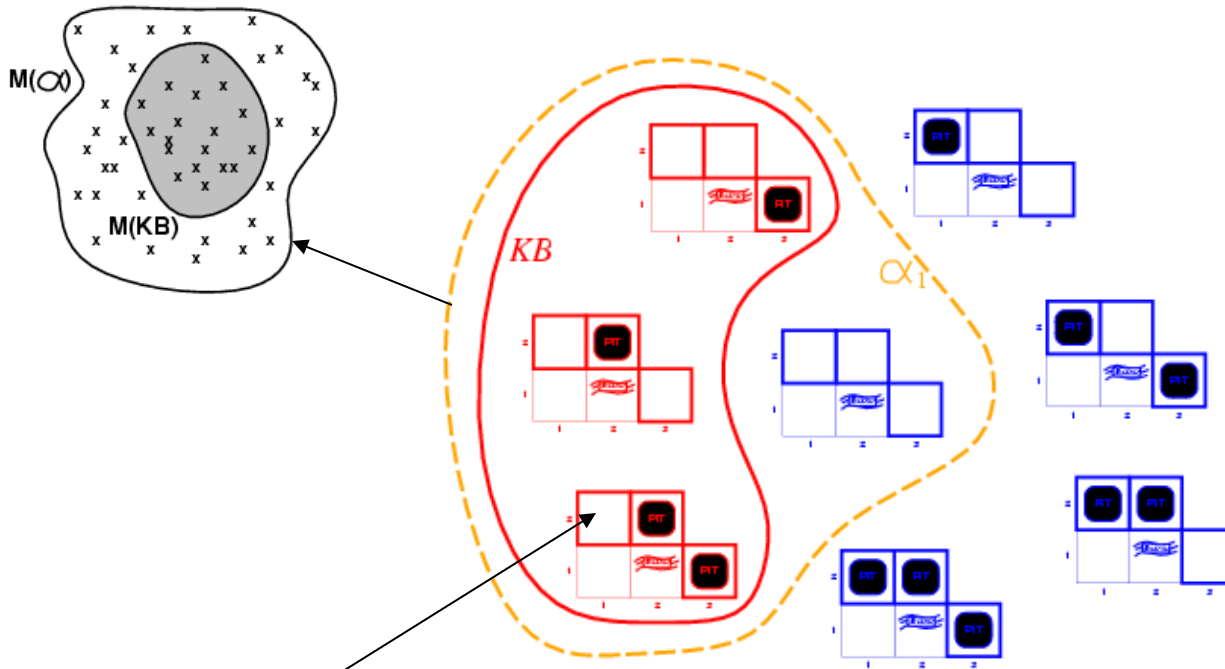


# Review: Wumpus models



- $KB$  = all possible wumpus-worlds consistent with the observations and the “physics” of the Wumpus world.

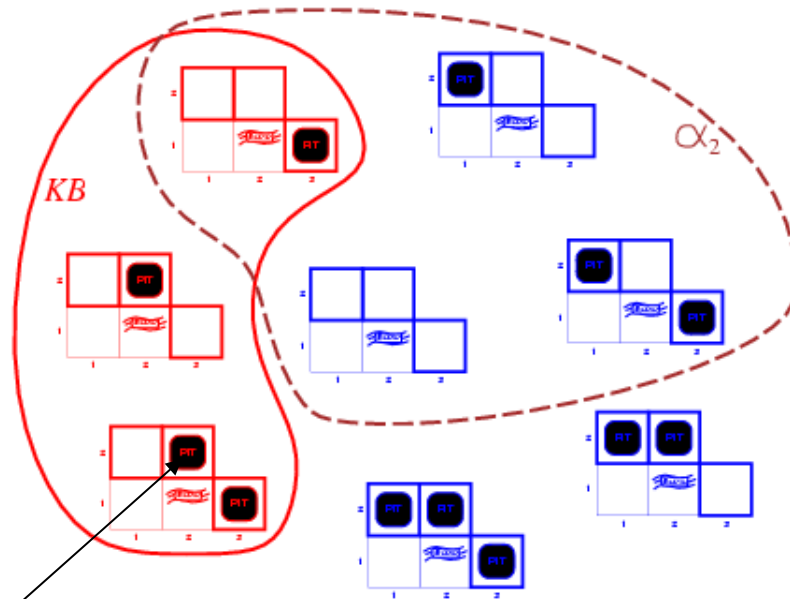
# Review: Wumpus models



$\alpha_1 = "[1,2] \text{ is safe}]", KB \models \alpha_1$ , proved by **model checking**.

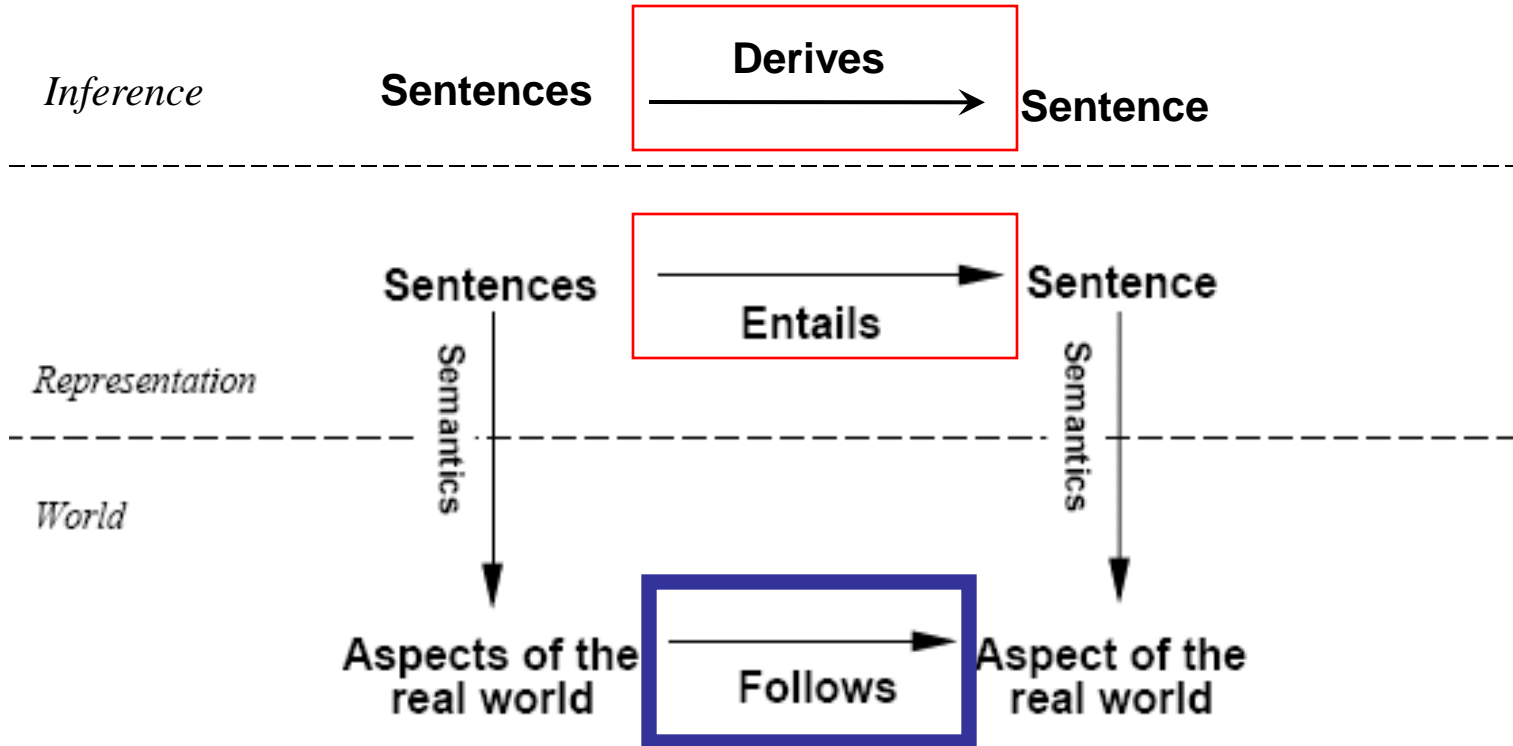
Every model that makes  $KB$  true also makes  $\alpha_1$  true.

# Wumpus models



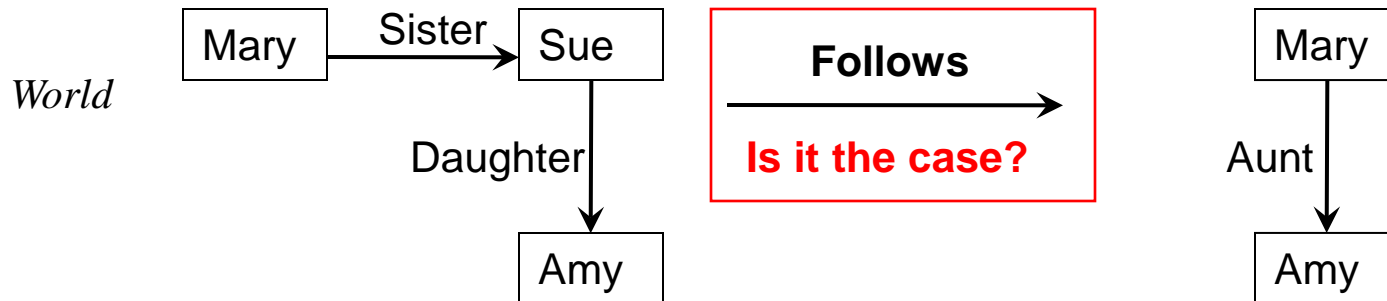
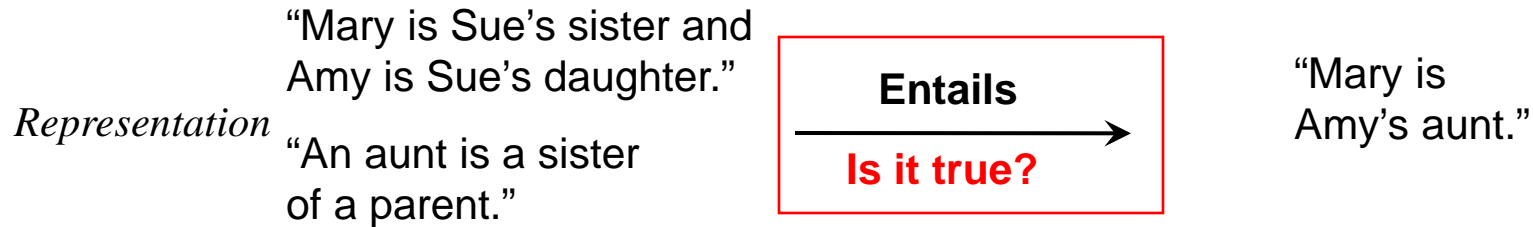
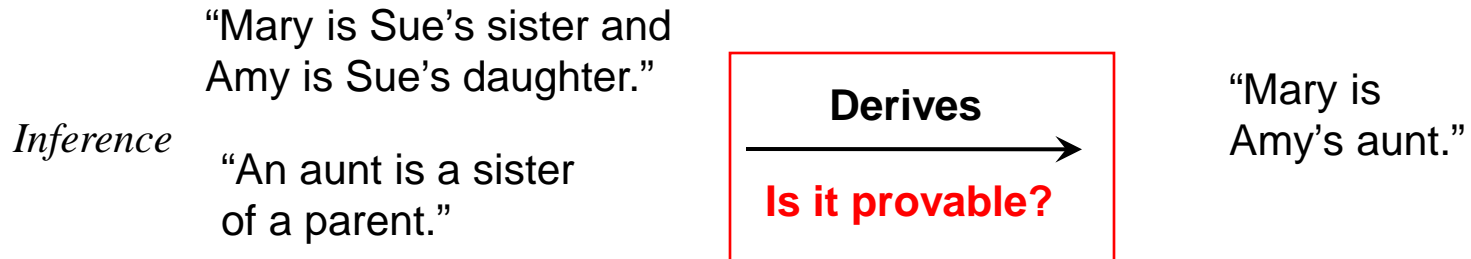
$\alpha_2 = "[2,2] \text{ is safe}", KB \neq \alpha_2$

# Review: Schematic for Follows, Entails, and Derives



*If KB is true in the real world,  
then any sentence  $\alpha$  **entailed** by KB  
and any sentence  $\alpha$  **derived** from KB  
**by a sound inference procedure**  
is also true in the real world.*

# Schematic Example: Follows, Entails, and Derives



# Recap propositional logic: Validity and satisfiability

A sentence is **valid** if it is true in **all** models,

e.g., *True*,  $A \vee \neg A$ ,  $A \Rightarrow A$ ,  $(A \wedge (A \Rightarrow B)) \Rightarrow B$

Validity is connected to inference via the **Deduction Theorem**:

$KB \models \alpha$  if and only if  $(KB \Rightarrow \alpha)$  is valid

A sentence is **satisfiable** if it is true in **some** model

e.g.,  $A \vee B$ ,  $C$

A sentence is **unsatisfiable** if it is **false** in **all** models

e.g.,  $A \wedge \neg A$

Satisfiability is connected to inference via the following:

$KB \models A$  if and only if  $(KB \wedge \neg A)$  is unsatisfiable

(there is no model for which KB is true and A is false)

# Inference Procedures

- $KB \vdash_i A$  means that sentence  $A$  can be derived from  $KB$  by procedure  $i$
- **Soundness**:  $i$  is sound if whenever  $KB \vdash_i \alpha$ , it is also true that  $KB \models \alpha$ 
  - *(no wrong inferences, but maybe not all inferences)*
- **Completeness**:  $i$  is complete if whenever  $KB \models \alpha$ , it is also true that  $KB \vdash_i \alpha$ 
  - *(all inferences can be made, but maybe some wrong extra ones as well)*
- Entailment can be used for inference (Model checking)
  - enumerate all possible models and check whether  $\alpha$  is true.
  - For  $n$  symbols, time complexity is  $O(2^n)$ ...
- Inference can be done directly on the sentences
  - Forward chaining, backward chaining, resolution (see FOPC, later)

# Resolution = Efficient Implication

Recall that  $(A \Rightarrow B) = ((\text{NOT } A) \text{ OR } B)$

and so:

$$\begin{aligned} (Y \text{ OR } X) &= ((\text{NOT } X) \Rightarrow Y) \\ ((\text{NOT } Y) \text{ OR } Z) &= (Y \Rightarrow Z) \end{aligned}$$

which yields:

$$((Y \text{ OR } X) \text{ AND } ((\text{NOT } Y) \text{ OR } Z)) = ((\text{NOT } X) \Rightarrow Z) = (X \text{ OR } Z)$$

(OR A B C D)

(OR  $\neg A$  E F G)

->Same ->

->Same ->

(NOT (OR B C D))  $\Rightarrow$  A

A  $\Rightarrow$  (OR E F G)

-----  
(OR B C D E F G)

-----  
(NOT (OR B C D))  $\Rightarrow$  (OR E F G)

-----  
(OR B C D E F G)

Recall: All clauses in KB are conjoined by an implicit AND (= CNF representation).

# Resolution Examples

- **Resolution:** inference rule for CNF: **sound and complete!** \*

$$(A \vee B \vee C)$$

$$(\neg A)$$

“If A or B or C is true, but not A, then B or C must be true.”

-----

$$\therefore (B \vee C)$$

$$(A \vee B \vee C)$$

$$(\neg A \vee D \vee E)$$

“If A is false then B or C must be true, or if A is true then D or E must be true, hence since A is either true or false, B or C or D or E must be true.”

-----

$$\therefore (B \vee C \vee D \vee E)$$

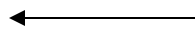
$$(A \vee B)$$

$$(\neg A \vee B)$$

“If A or B is true, and not A or B is true, then B must be true.”

-----

$$\therefore (B \vee B) \equiv B$$



Simplification  
is done always.

\* Resolution is “refutation complete” in that it can prove the truth of any entailed sentence by refutation.

# Only Resolve ONE Literal Pair!

If more than one pair, result always = TRUE.

Useless!! Always simplifies to TRUE!!

**No!**

(OR A B C D)  
(OR  $\neg A$   $\neg B$  F G)

---

(OR C D F G)

**No! This is wrong!**

**No!**

(OR A B C D)  
(OR  $\neg A$   $\neg B$   $\neg C$  )

---

(OR D)

**No! This is wrong!**

**Yes! (but = TRUE)**

(OR A B C D)  
(OR  $\neg A$   $\neg B$  F G)

---

(OR B  $\neg B$  C D F G)

**Yes! (but = TRUE)**

**Yes! (but = TRUE)**

(OR A B C D)  
(OR  $\neg A$   $\neg B$   $\neg C$  )

---

(OR A  $\neg A$  B  $\neg B$  D)

**Yes! (but = TRUE)**

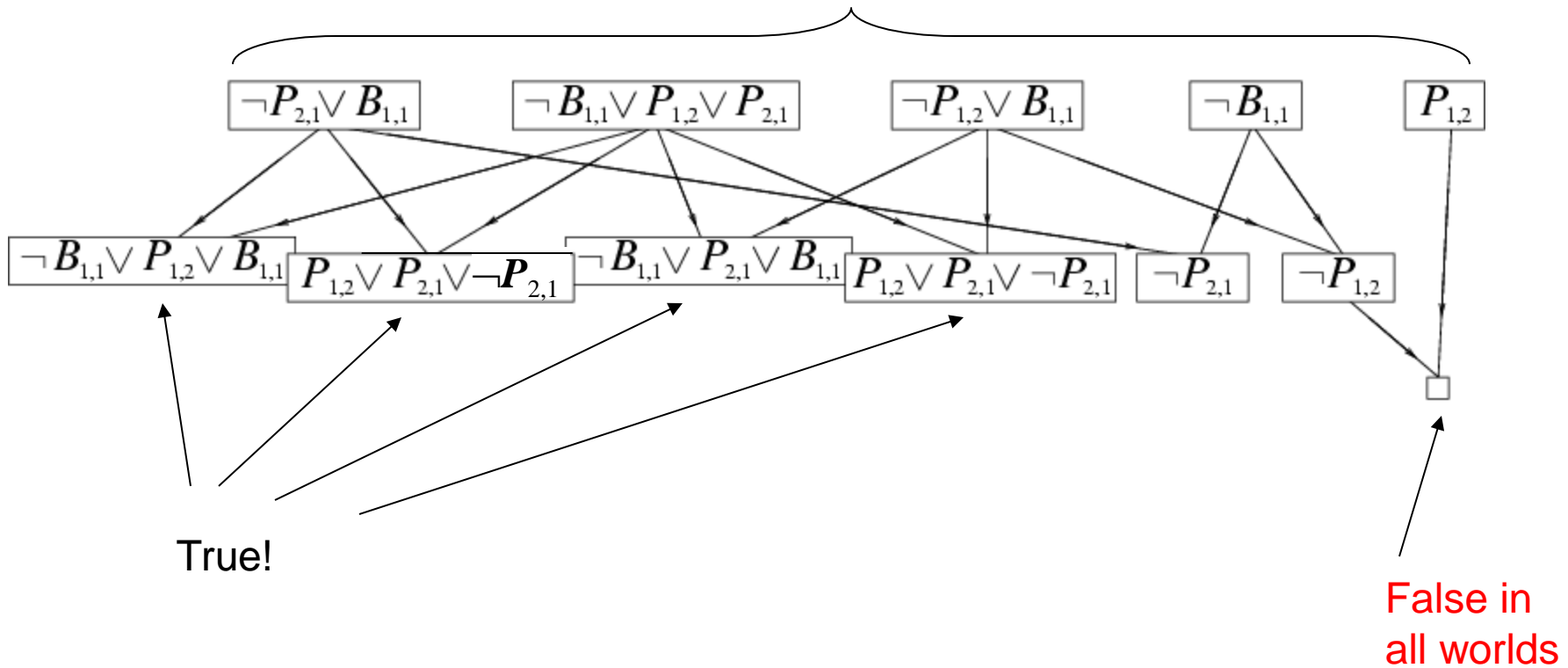
# Resolution Algorithm

- The resolution algorithm tries to prove:  $KB \models \alpha$  *equivalent to*  
 $KB \wedge \neg \alpha$  *unsatisfiable*
- Generate all new sentences from KB and the (negated) query.
- One of two things can happen:
  1. We find  $P \wedge \neg P$  which is unsatisfiable. I.e. we can entail the query.
  2. We find no contradiction: there is a model that satisfies the sentence  $KB \wedge \neg \alpha$  (non-trivial) and hence we cannot entail the query.

# Resolution example

- $KB = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$
- $\alpha = \neg P_{1,2}$

$KB \wedge \neg \alpha$



# Detailed Resolution Proof Example

- **In words:** *If the unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical if it is horned.*

Prove that the unicorn is both magical and horned.

$(\neg Y) (\neg R)$	$(M Y)$	$(R Y)$	$(H (\neg M))$
$(H R)$	$(\neg H) G$	$(\neg G) (\neg H)$	

- **Fourth, produce a resolution proof ending in ( ):**
- Resolve  $(\neg H \neg G)$  and  $(\neg H G)$  to give  $(\neg H)$
- Resolve  $(\neg Y \neg R)$  and  $(Y M)$  to give  $(\neg R M)$
- Resolve  $(\neg R M)$  and  $(R H)$  to give  $(M H)$
- Resolve  $(M H)$  and  $(\neg M H)$  to give  $(H)$
- Resolve  $(\neg H)$  and  $(H)$  to give  $( )$
- Of course, there are many other proofs, which are OK iff correct.

# Propositional Logic --- Summary

- Logical agents apply inference to a knowledge base to derive new information and make decisions
- Basic concepts of logic:
  - syntax: formal structure of sentences
  - semantics: truth of sentences wrt models
  - entailment: necessary truth of one sentence given another
  - inference: deriving sentences from other sentences
  - soundness: derivations produce only entailed sentences
  - completeness: derivations can produce all entailed sentences
  - valid: sentence is true in every model (a tautology)
- Logical equivalences allow syntactic manipulations
- Propositional logic lacks expressive power
  - Can only state specific facts about the world.
  - Cannot express general rules about the world  
(use First Order Predicate Logic instead)

# CS-171 Final Review

- **Propositional Logic**
  - (7.1-7.5)
- **First-Order Logic, Knowledge Representation**
  - (8.1-8.5, 9.1-9.2)
- **Probability & Bayesian Networks**
  - (13, 14.1-14.5)
- **Machine Learning**
  - (18.1-18.4)
- Questions on any topic
- Pre-mid-term material if time and class interest
- Please review your quizzes, mid-term, & old tests
  - At least one question from a prior quiz or old CS-171 test will appear on the Final Exam (and all other tests)

# Knowledge Representation using First-Order Logic

---

- Propositional Logic is **Useful** --- but has **Limited Expressive Power**
- First Order Predicate Calculus (FOPC), or First Order Logic (FOL).
  - FOPC has greatly expanded expressive power, though still limited.
- New Ontology
  - The world consists of OBJECTS (for propositional logic, the world was facts).
  - OBJECTS have PROPERTIES and engage in RELATIONS and FUNCTIONS.
- New Syntax
  - Constants, Predicates, Functions, Properties, Quantifiers.
- New Semantics
  - Meaning of new syntax.
- Knowledge engineering in FOL

## Review: Syntax of FOL: Basic elements

---

- Constants KingJohn, 2, UCI,...
- Predicates Brother, >,...
- Functions Sqrt, LeftLegOf,...
- Variables  $x, y, a, b, \dots$
- Connectives  $\neg, \Rightarrow, \wedge, \vee, \Leftrightarrow$
- Equality  $=$
- Quantifiers  $\forall, \exists$

# Syntax of FOL: Basic syntax elements are symbols

---

- **Constant Symbols:**
  - Stand for objects in the world.
    - E.g., KingJohn, 2, UCI, ...
- **Predicate Symbols**
  - Stand for relations (maps a tuple of objects to a **truth-value**)
    - E.g., Brother(Richard, John), greater\_than(3,2), ...
  - $P(x, y)$  is usually read as “x is P of y.”
    - E.g., Mother(Ann, Sue) is usually “Ann is Mother of Sue.”
- **Function Symbols**
  - Stand for functions (maps a tuple of objects to an **object**)
    - E.g., Sqrt(3), LeftLegOf(John), ...
- **Model** (world) = set of domain objects, relations, functions
- **Interpretation** maps symbols onto the model (world)
  - Very many interpretations are possible for each KB and world!
  - Job of the KB is to rule out models inconsistent with our knowledge.

# Syntax of FOL: Terms

---

- **Term** = logical expression that **refers to an object**
- **There are two kinds of terms:**
  - **Constant Symbols** stand for (or name) objects:
    - E.g., KingJohn, 2, UCI, Wumpus, ...
  - **Function Symbols** map tuples of objects to an object:
    - E.g., LeftLeg(KingJohn), Mother(Mary), Sqrt(x)
    - This is nothing but a complicated kind of name
      - No “subroutine” call, no “return value”

# Syntax of FOL: Atomic Sentences

---

- **Atomic Sentences** state facts (logical truth values).
  - An **atomic sentence** is a Predicate symbol, optionally followed by a parenthesized list of any argument terms
  - E.g., *Married( Father(Richard), Mother(John) )*
  - An **atomic sentence** asserts that some relationship (some predicate) holds among the objects that are its arguments.
- An **Atomic Sentence is true** in a given model if the relation referred to by the predicate symbol holds among the objects (terms) referred to by the arguments.

# Syntax of FOL: Connectives & Complex Sentences

---

- **Complex Sentences** are formed in the same way, and are formed using the same logical connectives, as we already know from propositional logic
- The **Logical Connectives**:
  - $\Leftrightarrow$  biconditional
  - $\Rightarrow$  implication
  - $\wedge$  and
  - $\vee$  or
  - $\neg$  negation
- **Semantics** for these logical connectives are the same as we already know from propositional logic.

# Syntax of FOL: Variables

---

- **Variables** range over objects in the world.
- A **variable** is like a **term** because it represents an object.
- A **variable** may be used wherever a **term** may be used.
  - **Variables** may be arguments to functions and predicates.
- A **term with NO variables** is called a **ground term**.
- All variables must be bound by a quantifier,  $\forall$  or  $\exists$
- (A **variable not bound by a quantifier** is called **free**.)
  - Used by mathematicians, not used in this class

# Syntax of FOL: Logical Quantifiers

---

- There are two **Logical Quantifiers**:
  - **Universal:**  $\forall x P(x)$  means "For all x, P(x)."
    - The "upside-down A" reminds you of "ALL."
  - **Existential:**  $\exists x P(x)$  means "There exists x such that, P(x)."
    - The "upside-down E" reminds you of "EXISTS."
- Syntactic "sugar" --- we really only need one quantifier.
  - $\forall x P(x) \equiv \neg \exists x \neg P(x)$
  - $\exists x P(x) \equiv \neg \forall x \neg P(x)$
  - You can ALWAYS convert one quantifier to the other.
- **RULES:**  $\forall \equiv \neg \exists \neg$  and  $\exists \equiv \neg \forall \neg$
- **RULE:** To move negation "in" across a quantifier, change the quantifier to "the other quantifier" and negate the predicate on "the other side."
  - $\neg \forall x P(x) \equiv \exists x \neg P(x)$
  - $\neg \exists x P(x) \equiv \forall x \neg P(x)$

# Universal Quantification $\forall$

---

- $\forall$  means “for all”
- Allows us to make statements about all objects that have certain properties
- Can now state general rules:

$\forall x \text{ King}(x) \Rightarrow \text{Person}(x)$  “All kings are persons.”

$\forall x \text{ Person}(x) \Rightarrow \text{HasHead}(x)$  “Every person has a head.”

$\forall i \text{ Integer}(i) \Rightarrow \text{Integer}(\text{plus}(i,1))$  “If  $i$  is an integer then  $i+1$  is an integer.”

Note that

$\forall x \text{ King}(x) \wedge \text{Person}(x)$  is not correct!

This would imply that all objects  $x$  are Kings and are People

$\forall x \text{ King}(x) \Rightarrow \text{Person}(x)$  is the correct way to say this

**Note that  $\Rightarrow$  is the natural connective to use with  $\forall$ .**

# Existential Quantification $\exists$

---

- $\exists x$  means “there exists an  $x$  such that....” (at least one object  $x$ )
- Allows us to make statements about some object without naming it
- Examples:

$\exists x \text{ King}(x)$  “Some object is a king.”

$\exists x \text{ Lives\_in}(\text{John}, \text{Castle}(x))$  “John lives in somebody’s castle.”

$\exists i \text{ Integer}(i) \wedge \text{GreaterThan}(i, 0)$  “Some integer is greater than zero.”

**Note that  $\wedge$  is the natural connective to use with  $\exists$**

(And remember that  $\Rightarrow$  is the natural connective to use with  $\forall$  )

## Combining Quantifiers --- Order (Scope)

---

The order of “unlike” quantifiers is important.

$\forall x \exists y \text{ Loves}(x,y)$

- For everyone (“all x”) there is someone (“exists y”) whom they love

$\exists y \forall x \text{ Loves}(x,y)$

- there is someone (“exists y”) whom everyone loves (“all x”)

Clearer with parentheses:  $\exists y ( \forall x \text{ Loves}(x,y) )$

The order of “like” quantifiers does not matter.

$\forall x \forall y P(x, y) \equiv \forall y \forall x P(x, y)$

$\exists x \exists y P(x, y) \equiv \exists y \exists x P(x, y)$

# De Morgan's Law for Quantifiers

---

De Morgan's Rule

$$P \wedge Q \equiv \neg(\neg P \vee \neg Q)$$

$$P \vee Q \equiv \neg(\neg P \wedge \neg Q)$$

$$\neg(P \wedge Q) \equiv \neg P \vee \neg Q$$

$$\neg(P \vee Q) \equiv \neg P \wedge \neg Q$$

Generalized De Morgan's Rule

$$\forall x P \equiv \neg \exists x (\neg P)$$

$$\exists x P \equiv \neg \forall x (\neg P)$$

$$\neg \forall x P \equiv \exists x (\neg P)$$

$$\neg \exists x P \equiv \forall x (\neg P)$$

Rule is simple: if you bring a negation inside a disjunction or a conjunction, always switch between them (or  $\rightarrow$  and, and  $\rightarrow$  or).

## Fun with sentences

Brothers are siblings

$$\forall x, y \text{ Brother}(x, y) \Rightarrow \text{Sibling}(x, y).$$

“Sibling” is symmetric

$$\forall x, y \text{ Sibling}(x, y) \Leftrightarrow \text{Sibling}(y, x).$$

One's mother is one's female parent

$$\forall x, y \text{ Mother}(x, y) \Leftrightarrow (\text{Female}(x) \wedge \text{Parent}(x, y)).$$

A first cousin is a child of a parent's sibling

$$\forall x, y \text{ FirstCousin}(x, y) \Leftrightarrow \exists p, ps \text{ Parent}(p, x) \wedge \text{Sibling}(ps, p) \wedge \text{Parent}(ps, y)$$

## More fun with sentences

---

- **“All persons are mortal.”**
- [Use:  $\text{Person}(x)$ ,  $\text{Mortal}(x)$  ]
- $\forall x \text{ Person}(x) \Rightarrow \text{Mortal}(x)$
- $\forall x \neg \text{Person}(x) \vee \text{Mortal}(x)$
- **Common Mistakes:**
- $\forall x \text{ Person}(x) \wedge \text{Mortal}(x)$
- **Note that  $\Rightarrow$  is the natural connective to use with  $\forall$ .**

## More fun with sentences

---

- “Fifi has a sister who is a cat.”
- [Use:  $\text{Sister}(\text{Fifi}, x), \text{Cat}(x)$  ]
- 
- $\exists x \text{ Sister}(\text{Fifi}, x) \wedge \text{Cat}(x)$
- **Common Mistakes:**
- $\exists x \text{ Sister}(\text{Fifi}, x) \Rightarrow \text{Cat}(x)$
- Note that  $\wedge$  is the natural connective to use with  $\exists$

## More fun with sentences

---

- **“For every food, there is a person who eats that food.”**
- [Use: Food(x), Person(y), Eats(y, x) ]
- All are correct:
  - $\forall x \exists y \text{ Food}(x) \Rightarrow [ \text{Person}(y) \wedge \text{Eats}(y, x) ]$
  - $\forall x \text{ Food}(x) \Rightarrow \exists y [ \text{Person}(y) \wedge \text{Eats}(y, x) ]$
  - $\forall x \exists y \neg \text{Food}(x) \vee [ \text{Person}(y) \wedge \text{Eats}(y, x) ]$
  - $\forall x \exists y [ \neg \text{Food}(x) \vee \text{Person}(y) ] \wedge [ \neg \text{Food}(x) \vee \text{Eats}(y, x) ]$
  - $\forall x \exists y [ \text{Food}(x) \Rightarrow \text{Person}(y) ] \wedge [ \text{Food}(x) \Rightarrow \text{Eats}(y, x) ]$
- **Common Mistakes:**
  - $\forall x \exists y [ \text{Food}(x) \wedge \text{Person}(y) ] \Rightarrow \text{Eats}(y, x)$
  - $\forall x \exists y \text{ Food}(x) \wedge \text{Person}(y) \wedge \text{Eats}(y, x)$

## More fun with sentences

---

- **"Every person eats every food."**
- [Use: Person (x), Food (y), Eats(x, y) ]
- 
- $\forall x \forall y [ \text{Person}(x) \wedge \text{Food}(y) ] \Rightarrow \text{Eats}(x, y)$
- $\forall x \forall y \neg \text{Person}(x) \vee \neg \text{Food}(y) \vee \text{Eats}(x, y)$
- $\forall x \forall y \text{Person}(x) \Rightarrow [ \text{Food}(y) \Rightarrow \text{Eats}(x, y) ]$
- $\forall x \forall y \text{Person}(x) \Rightarrow [ \neg \text{Food}(y) \vee \text{Eats}(x, y) ]$
- $\forall x \forall y \neg \text{Person}(x) \vee [ \text{Food}(y) \Rightarrow \text{Eats}(x, y) ]$
- **Common Mistakes:**
- $\forall x \forall y \text{Person}(x) \Rightarrow [ \text{Food}(y) \wedge \text{Eats}(x, y) ]$
- $\forall x \forall y \text{Person}(x) \wedge \text{Food}(y) \wedge \text{Eats}(x, y)$

## More fun with sentences

---

- **“All greedy kings are evil.”**
- [Use: King(x), Greedy(x), Evil(x) ]
- 
- $\forall x [ \text{Greedy}(x) \wedge \text{King}(x) ] \Rightarrow \text{Evil}(x)$
- $\forall x \neg \text{Greedy}(x) \vee \neg \text{King}(x) \vee \text{Evil}(x)$
- $\forall x \text{Greedy}(x) \Rightarrow [ \text{King}(x) \Rightarrow \text{Evil}(x) ]$
- **Common Mistakes:**
- $\forall x \text{Greedy}(x) \wedge \text{King}(x) \wedge \text{Evil}(x)$

## More fun with sentences

---

- **"Everyone has a favorite food."**
- [Use: Person(x), Food(y), Favorite(y, x) ]
- 
- $\forall x \exists y \text{ Person}(x) \Rightarrow [ \text{Food}(y) \wedge \text{Favorite}(y, x) ]$
- $\forall x \text{ Person}(x) \Rightarrow \exists y [ \text{Food}(y) \wedge \text{Favorite}(y, x) ]$
- $\forall x \exists y \neg \text{Person}(x) \vee [ \text{Food}(y) \wedge \text{Favorite}(y, x) ]$
- $\forall x \exists y [ \neg \text{Person}(x) \vee \text{Food}(y) ] \wedge [ \neg \text{Person}(x) \vee \text{Favorite}(y, x) ]$
- $\forall x \exists y [ \text{Person}(x) \Rightarrow \text{Food}(y) ] \wedge [ \text{Person}(x) \Rightarrow \text{Favorite}(y, x) ]$
- **Common Mistakes:**
- $\forall x \exists y [ \text{Person}(x) \wedge \text{Food}(y) ] \Rightarrow \text{Favorite}(y, x)$
- $\forall x \exists y \text{ Person}(x) \wedge \text{Food}(y) \wedge \text{Favorite}(y, x)$

# Semantics: Interpretation

---

- An **interpretation** of a sentence (wff) is an assignment that maps
  - Object constant symbols to objects in the world,
  - n-ary function symbols to n-ary functions in the world,
  - n-ary relation symbols to n-ary relations in the world
- Given an interpretation, an atomic sentence has the value “true” if it denotes a relation that holds for those individuals denoted in the terms. Otherwise it has the value “false.”
  - Example: Kinship world:
    - Symbols = Ann, Bill, Sue, Married, Parent, Child, Sibling, ...
  - World consists of individuals in relations:
    - Married(Ann,Bill) is false, Parent(Bill,Sue) is true, ...
- Your job, as a Knowledge Engineer, is to construct KB so it is true **\*exactly\*** for your world and intended interpretation.

# Semantics: Models and Definitions

---

- An interpretation and possible world satisfies a wff (sentence) if the wff has the value “true” under that interpretation in that possible world.
- A domain and an interpretation that satisfies a wff is a model of that wff
- Any wff that has the value “true” in all possible worlds and under all interpretations is valid.
- Any wff that does not have a model under any interpretation is inconsistent or unsatisfiable.
- Any wff that is true in at least one possible world under at least one interpretation is satisfiable.
- If a wff  $w$  has a value true under all the models of a set of sentences  $KB$  then  $KB$  logically entails  $w$ .

## Conversion to CNF

---

- Everyone who loves all animals is loved by someone:

$$\forall x [\forall y \text{ Animal}(y) \Rightarrow \text{Loves}(x,y)] \Rightarrow [\exists y \text{ Loves}(y,x)]$$

1. Eliminate biconditionals and implications

$$\forall x [\neg \forall y \neg \text{Animal}(y) \vee \text{Loves}(x,y)] \vee [\exists y \text{ Loves}(y,x)]$$

2. Move  $\neg$  inwards:

$$\neg \forall x p \equiv \exists x \neg p, \quad \neg \exists x p \equiv \forall x \neg p$$

$$\forall x [\exists y \neg(\neg \text{Animal}(y) \vee \text{Loves}(x,y))] \vee [\exists y \text{ Loves}(y,x)]$$

$$\forall x [\exists y \neg \neg \text{Animal}(y) \wedge \neg \text{Loves}(x,y)] \vee [\exists y \text{ Loves}(y,x)]$$

$$\forall x [\exists y \text{ Animal}(y) \wedge \neg \text{Loves}(x,y)] \vee [\exists y \text{ Loves}(y,x)]$$

## Conversion to CNF contd.

---

3. Standardize variables: each quantifier should use a different one

$$\forall x [\exists y \textit{Animal}(y) \wedge \neg \textit{Loves}(x,y)] \vee [\exists z \textit{Loves}(z,x)]$$

4. Skolemize: a more general form of existential instantiation. Each existential variable is replaced by a **Skolem function** of the enclosing universally quantified variables:

$$\forall x [\textit{Animal}(F(x)) \wedge \neg \textit{Loves}(x,F(x))] \vee \textit{Loves}(G(x),x)$$

5. Drop universal quantifiers:  
 $[\textit{Animal}(F(x)) \wedge \neg \textit{Loves}(x,F(x))] \vee \textit{Loves}(G(x),x)$

6. Distribute  $\vee$  over  $\wedge$  :  
 $[\textit{Animal}(F(x)) \vee \textit{Loves}(G(x),x)] \wedge [\neg \textit{Loves}(x,F(x)) \vee \textit{Loves}(G(x),x)]$

# Unification

---

- Recall:  $\text{Subst}(\theta, p)$  = result of substituting  $\theta$  into sentence  $p$
- Unify algorithm: takes 2 sentences  $p$  and  $q$  and returns a unifier if one exists

$$\text{Unify}(p, q) = \theta \quad \text{where } \text{Subst}(\theta, p) = \text{Subst}(\theta, q)$$

- Example:  
     $p = \text{Knows}(\text{John}, x)$   
     $q = \text{Knows}(\text{John}, \text{Jane})$

$$\text{Unify}(p, q) = \{x/\text{Jane}\}$$

# Unification examples

---

- simple example: query =  $\text{Knows}(\text{John}, x)$ , i.e., who does John know?

p	q	$\theta$
$\text{Knows}(\text{John}, x)$	$\text{Knows}(\text{John}, \text{Jane})$	$\{x/\text{Jane}\}$
$\text{Knows}(\text{John}, x)$	$\text{Knows}(y, \text{OJ})$	$\{x/\text{OJ}, y/\text{John}\}$
$\text{Knows}(\text{John}, x)$	$\text{Knows}(y, \text{Mother}(y))$	$\{y/\text{John}, x/\text{Mother}(\text{John})\}$
$\text{Knows}(\text{John}, x)$	$\text{Knows}(x, \text{OJ})$	$\{\text{fail}\}$

- Last unification fails: only because  $x$  can't take values John and OJ at the same time
  - But we know that if John knows  $x$ , and everyone ( $x$ ) knows OJ, we should be able to infer that John knows OJ
- Problem is due to use of same variable  $x$  in both sentences
- Simple solution: Standardizing apart eliminates overlap of variables, e.g.,  $\text{Knows}(z, \text{OJ})$

# Unification

---

- To unify  $Knows(John, x)$  and  $Knows(y, z)$ ,

$$\theta = \{y/John, x/z\} \text{ or } \theta = \{y/John, x/John, z/John\}$$

- The first unifier is **more general** than the second.
- There is a single **most general unifier** (MGU) that is unique up to renaming of variables.

$$MGU = \{y/John, x/z\}$$

- General algorithm in Figure 9.1 in the text

# Unification Algorithm

```
function UNIFY( $x, y, \theta$ ) returns a substitution to make  $x$  and  $y$  identical
  inputs:  $x$ , a variable, constant, list, or compound expression
            $y$ , a variable, constant, list, or compound expression
            $\theta$ , the substitution built up so far (optional, defaults to empty)

  if  $\theta = \text{failure}$  then return failure
  else if  $x = y$  then return  $\theta$ 
  else if VARIABLE?( $x$ ) then return UNIFY-VAR( $x, y, \theta$ )
  else if VARIABLE?( $y$ ) then return UNIFY-VAR( $y, x, \theta$ )
  else if COMPOUND?( $x$ ) and COMPOUND?( $y$ ) then
    return UNIFY( $x$ .ARGS,  $y$ .ARGS, UNIFY( $x$ .OP,  $y$ .OP,  $\theta$ ))
  else if LIST?( $x$ ) and LIST?( $y$ ) then
    return UNIFY( $x$ .REST,  $y$ .REST, UNIFY( $x$ .FIRST,  $y$ .FIRST,  $\theta$ ))
  else return failure
```

```
function UNIFY-VAR( $var, x, \theta$ ) returns a substitution

  if  $\{var/val\} \in \theta$  then return UNIFY( $val, x, \theta$ )
  else if  $\{x/val\} \in \theta$  then return UNIFY( $var, val, \theta$ )
  else if OCCUR-CHECK?( $var, x$ ) then return failure
  else return add  $\{var/x\}$  to  $\theta$ 
```

**Figure 9.1** The unification algorithm. The algorithm works by comparing the structures of the inputs, element by element. The substitution  $\theta$  that is the argument to UNIFY is built up along the way and is used to make sure that later comparisons are consistent with bindings that were established earlier. In a compound expression such as  $F(A, B)$ , the OP field picks out the function symbol  $F$  and the ARGS field picks out the argument list  $(A, B)$ .

# Knowledge engineering in FOL

---

1. Identify the task
2. Assemble the relevant knowledge
3. Decide on a vocabulary of predicates, functions, and constants
4. Encode general knowledge about the domain
5. Encode a description of the specific problem instance
6. Pose queries to the inference procedure and get answers
7. Debug the knowledge base

# The electronic circuits domain

---

1. Identify the task
  - Does the circuit actually add properly?
2. Assemble the relevant knowledge
  - Composed of wires and gates; Types of gates (AND, OR, XOR, NOT)
  - 
  - Irrelevant: size, shape, color, cost of gates
  -
3. Decide on a vocabulary
  - Alternatives:
  - - Type( $X_1$ ) = XOR (function)
    - Type( $X_1$ , XOR) (binary predicate)
    - XOR( $X_1$ )
    - (unary predicate)

# The electronic circuits domain

---

4. Encode general knowledge of the domain
- $\forall t_1, t_2 \text{ Connected}(t_1, t_2) \Rightarrow \text{Signal}(t_1) = \text{Signal}(t_2)$
  - $\forall t \text{ Signal}(t) = 1 \vee \text{Signal}(t) = 0$
  - $1 \neq 0$
  - $\forall t_1, t_2 \text{ Connected}(t_1, t_2) \Rightarrow \text{Connected}(t_2, t_1)$
  - $\forall g \text{ Type}(g) = \text{OR} \Rightarrow \text{Signal}(\text{Out}(1, g)) = 1 \Leftrightarrow \exists n \text{ Signal}(\text{In}(n, g)) = 1$
  - $\forall g \text{ Type}(g) = \text{AND} \Rightarrow \text{Signal}(\text{Out}(1, g)) = 0 \Leftrightarrow \exists n \text{ Signal}(\text{In}(n, g)) = 0$
  - $\forall g \text{ Type}(g) = \text{XOR} \Rightarrow \text{Signal}(\text{Out}(1, g)) = 1 \Leftrightarrow \text{Signal}(\text{In}(1, g)) \neq \text{Signal}(\text{In}(2, g))$
  - $\forall g \text{ Type}(g) = \text{NOT} \Rightarrow \text{Signal}(\text{Out}(1, g)) \neq \text{Signal}(\text{In}(1, g))$

# The electronic circuits domain

---

## 5. Encode the specific problem instance

Type( $X_1$ ) = XOR

Type( $A_1$ ) = AND

Type( $O_1$ ) = OR

Type( $X_2$ ) = XOR

Type( $A_2$ ) = AND

Connected(Out(1, $X_1$ ),In(1, $X_2$ ))

Connected(Out(1, $X_1$ ),In(2, $A_2$ ))

Connected(Out(1, $A_2$ ),In(1, $O_1$ ))

Connected(Out(1, $A_1$ ),In(2, $O_1$ ))

Connected(Out(1, $X_2$ ),Out(1, $C_1$ ))

Connected(Out(1, $O_1$ ),Out(2, $C_1$ ))

Connected(In(1, $C_1$ ),In(1, $X_1$ ))

Connected(In(1, $C_1$ ),In(1, $A_1$ ))

Connected(In(2, $C_1$ ),In(2, $X_1$ ))

Connected(In(2, $C_1$ ),In(2, $A_1$ ))

Connected(In(3, $C_1$ ),In(2, $X_2$ ))

Connected(In(3, $C_1$ ),In(1, $A_2$ ))

# The electronic circuits domain

---

## 6. Pose queries to the inference procedure

What are the possible sets of values of all the terminals for the adder circuit?

$$\exists i_1, i_2, i_3, o_1, o_2 \text{ Signal(In}(1, C_1)) = i_1 \wedge \text{Signal(In}(2, C_1)) = i_2 \wedge \text{Signal(In}(3, C_1)) = i_3 \wedge \text{Signal(Out}(1, C_1)) = o_1 \wedge \text{Signal(Out}(2, C_1)) = o_2$$

## 7. Debug the knowledge base

May have omitted assertions like  $1 \neq 0$

# CS-171 Final Review

---

- **Propositional Logic**
  - (7.1-7.5)
- **First-Order Logic, Knowledge Representation**
  - (8.1-8.5, 9.1-9.2)
- **Probability & Bayesian Networks**
  - (13, 14.1-14.5)
- **Machine Learning**
  - (18.1-18.4)
- Questions on any topic
- Pre-mid-term material if time and class interest
- Please review your quizzes, mid-term, & old tests
  - At least one question from a prior quiz or old CS-171 test will appear on the Final Exam (and all other tests)

# You will be expected to know

- Basic probability notation/definitions:
  - Probability model, unconditional/prior and conditional/posterior probabilities, factored representation (= variable/value pairs), random variable, (joint) probability distribution, probability density function (pdf), marginal probability, (conditional) independence, normalization, etc.
- Basic probability formulae:
  - Probability axioms, sum rule, product rule, Bayes' rule.
- How to use Bayes' rule:
  - Naïve Bayes model (naïve Bayes classifier)

# Syntax

- Basic element: **random variable**
- Similar to propositional logic: possible worlds defined by assignment of values to random variables.
- **Boolean** random variables  
e.g., *Cavity (= do I have a cavity?)*
- **Discrete** random variables  
e.g., *Weather is one of <sunny,rainy,cloudy,snow>*
- Domain values must be exhaustive and mutually exclusive
- Elementary proposition is an assignment of a value to a random variable:  
e.g., *Weather = sunny; Cavity = false (abbreviated as  $\neg$ cavity)*
- Complex propositions formed from elementary propositions and standard logical connectives :  
e.g., *Weather = sunny  $\vee$  Cavity = false*

# Probability

- $P(a)$  is the probability of proposition “a”
  - e.g.,  $P(\text{it will rain in London tomorrow})$
  - The proposition  $a$  is actually true or false in the real-world
- **Probability Axioms:**
  - $0 \leq P(a) \leq 1$
  - $P(\text{NOT}(a)) = 1 - P(a)$                        $\Rightarrow \quad \sum_A P(A) = 1$
  - $P(\text{true}) = 1$
  - $P(\text{false}) = 0$
  - $P(A \text{ OR } B) = P(A) + P(B) - P(A \text{ AND } B)$
- Any agent that holds degrees of beliefs that contradict these axioms will act irrationally in some cases
- **Rational agents cannot violate probability theory.**
  - Acting otherwise results in irrational behavior.

# Conditional Probability

- $P(a|b)$  is the conditional probability of proposition  $a$ , conditioned on knowing that  $b$  is true,
  - E.g.,  $P(\text{rain in London tomorrow} \mid \text{raining in London today})$
  - $P(a|b)$  is a “posterior” or conditional probability
  - The updated probability that  $a$  is true, now that we know  $b$
  - $P(a|b) = P(a \wedge b) / P(b)$
  - Syntax:  $P(a \mid b)$  is the probability of  $a$  given that  $b$  is true
    - $a$  and  $b$  can be any propositional sentences
    - e.g.,  $p(\text{John wins OR Mary wins} \mid \text{Bob wins AND Jack loses})$
- $P(a|b)$  obeys the same rules as probabilities,
  - E.g.,  $P(a \mid b) + P(\text{NOT}(a) \mid b) = 1$
  - All probabilities in effect are conditional probabilities
    - E.g.,  $P(a) = P(a \mid \text{our background knowledge})$

# Concepts of Probability

- **Unconditional Probability**

- **$P(a)$** , the probability of “a” being true, or  **$P(a=\text{True})$**
- Does not depend on anything else to be true (**unconditional**)
- Represents the probability prior to further information that may adjust it (**prior**)

- **Conditional Probability**


- **$P(a|b)$** , the probability of “a” being true, given that “b” is true
- Relies on “b” = true (**conditional**)
- Represents the prior probability adjusted based upon new information “b” (**posterior**)
- Can be generalized to more than 2 random variables:
  - e.g.  $P(a|b, c, d)$

- **Joint Probability**

- **$P(a, b) = P(a \wedge b)$** , the probability of “a” and “b” both being true
- Can be generalized to more than 2 random variables:
  - e.g.  $P(a, b, c, d)$

# Basic Probability Relationships

- **$P(A) + P(\neg A) = 1$** 
  - Implies that  $P(\neg A) = 1 - P(A)$
- **$P(A, B) = P(A \wedge B) = P(A) + P(B) - P(A \vee B)$** 
  - Implies that  $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$
- **$P(A | B) = P(A, B) / P(B)$** 
  - Conditional probability; “Probability of A given B”
- **$P(A, B) = P(A | B) P(B)$** 
  - Product Rule (Factoring); applies to any number of variables
  - $P(a, b, c, \dots, z) = P(a | b, c, \dots, z) P(b | c, \dots, z) P(c | \dots, z) \dots P(z)$
- **$P(A) = \sum_{B, C} P(A, B, C) = \sum_{b \in B, c \in C} P(A, b, c)$** 
  - Sum Rule (Marginal Probabilities); for any number of variables
  - $P(A, D) = \sum_B \sum_C P(A, B, C, D) = \sum_{b \in B} \sum_{c \in C} P(A, b, c, D)$
- **$P(B | A) = P(A | B) P(B) / P(A)$** 
  - Bayes’ Rule; for any number of variables



You need to know these !

# Summary of Probability Rules

- **Product Rule:**

- $P(a, b) = P(a|b) P(b) = P(b|a) P(a)$
- Probability of “a” and “b” occurring is the same as probability of “a” occurring given “b” is true, times the probability of “b” occurring.
  - e.g.,  $P(\text{rain, cloudy}) = P(\text{rain} | \text{cloudy}) * P(\text{cloudy})$

- **Sum Rule:** (AKA **Law of Total Probability**)

- $P(a) = \sum_b P(a, b) = \sum_b P(a|b) P(b)$ , where B is any random variable
- Probability of “a” occurring is the same as the sum of all joint probabilities including the event, provided the joint probabilities represent all possible events.
- Can be used to “marginalize” out other variables from probabilities, resulting in prior probabilities also being called marginal probabilities.
  - e.g.,  $P(\text{rain}) = \sum_{\text{Windspeed}} P(\text{rain, Windspeed})$   
where  $\text{Windspeed} = \{0\text{-}10\text{mph}, 10\text{-}20\text{mph}, 20\text{-}30\text{mph}, \text{etc.}\}$

- **Bayes’ Rule:**

- $P(b|a) = P(a|b) P(b) / P(a)$
- Acquired from rearranging the product rule.
- Allows conversion between conditionals, from  $P(a|b)$  to  $P(b|a)$ .
  - e.g.,  $b = \text{disease}, a = \text{symptoms}$   
More natural to encode knowledge as  $P(a|b)$  than as  $P(b|a)$ .

# Full Joint Distribution

- We can fully specify a probability space by constructing a **full joint distribution**:
  - A full joint distribution contains a probability for every possible combination of variable values.
  - E.g.,  $P(J=f, M=t, A=t, B=t, E=f)$
- From a full joint distribution, the product rule, sum rule, and Bayes' rule can create any desired joint and conditional probabilities.

# Computing with Probabilities: Law of Total Probability

Law of Total Probability (aka “summing out” or marginalization)

$$\begin{aligned} P(a) &= \sum_b P(a, b) \\ &= \sum_b P(a \mid b) P(b) \quad \text{where } B \text{ is any random variable} \end{aligned}$$

Why is this useful?

Given a joint distribution (e.g.,  $P(a,b,c,d)$ ) we can obtain any “marginal” probability (e.g.,  $P(b)$ ) by summing out the other variables, e.g.,

$$P(b) = \sum_a \sum_c \sum_d P(a, b, c, d)$$

We can compute any conditional probability given a joint distribution, e.g.,

$$\begin{aligned} P(c \mid b) &= \sum_a \sum_d P(a, c, d \mid b) \\ &= \sum_a \sum_d P(a, c, d, b) / P(b) \\ &\quad \text{where } P(b) \text{ can be computed as above} \end{aligned}$$

# Computing with Probabilities: The Chain Rule or Factoring

We can always write

$$P(a, b, c, \dots z) = P(a \mid b, c, \dots z) P(b, c, \dots z)$$

(by definition of joint probability)

Repeatedly applying this idea, we can write

$$P(a, b, c, \dots z) = P(a \mid b, c, \dots z) P(b \mid c, \dots z) P(c \mid \dots z) \dots P(z)$$

This factorization holds for any ordering of the variables

This is the chain rule for probabilities

# Independence

- Formal Definition:
  - 2 random variables A and B are **independent** iff:  
$$P(a, b) = P(a) P(b), \quad \text{for all values } a, b$$
- Informal Definition:
  - 2 random variables A and B are **independent** iff:  
$$P(a | b) = P(a) \quad \text{OR} \quad P(b | a) = P(b), \quad \text{for all values } a, b$$
  - $P(a | b) = P(a)$  tells us that knowing b provides no change in our probability for a, and thus b contains no information about a.
- Also known as **marginal independence**, as all other variables have been marginalized out.
- In practice true independence is very rare:
  - “butterfly in China” effect
  - Conditional independence is much more common and useful

# Conditional Independence

- Formal Definition:

- 2 random variables A and B are **conditionally independent** given C iff:

$$P(a, b | c) = P(a | c) P(b | c), \quad \text{for all values } a, b, c$$

- Informal Definition:

- 2 random variables A and B are **conditionally independent** given C iff:

$$P(a | b, c) = P(a | c) \quad \text{OR} \quad P(b | a, c) = P(b | c), \quad \text{for all values } a, b, c$$

- $P(a | b, c) = P(a | c)$  tells us that learning about b, given that we already know c, provides no change in our probability for a, and thus b contains no information about a beyond what c provides.

- Naïve Bayes Model:

- Often a single variable can directly influence a number of other variables, all of which are conditionally independent, given the single variable.
- E.g., k different symptom variables  $X_1, X_2, \dots, X_k$ , and  $C$  = disease, reducing to:

$$P(X_1, X_2, \dots, X_k | C) = P(C) \prod P(X_i | C)$$

# Examples of Conditional Independence

- **H=Heat, S=Smoke, F=Fire**

- $P(H, S \mid F) = P(H \mid F) P(S \mid F)$
- $P(S \mid F, S) = P(S \mid F)$
- If we know there is/is not a fire, observing heat tells us no more information about smoke

- **F=Fever, R=RedSpots, M=Measles**

- $P(F, R \mid M) = P(F \mid M) P(R \mid M)$
- $P(R \mid M, F) = P(R \mid M)$
- If we know we do/don't have measles, observing fever tells us no more information about red spots

- **C=SharpClaws, F=SharpFangs, S=Species**

- $P(C, F \mid S) = P(C \mid S) P(F \mid S)$
- $P(F \mid S, C) = P(F \mid S)$
- If we know the species, observing sharp claws tells us no more information about sharp fangs

# CS-171 Final Review

- **Propositional Logic**
  - (7.1-7.5)
- **First-Order Logic, Knowledge Representation**
  - (8.1-8.5, 9.1-9.2)
- **Probability & Bayesian Networks**
  - (13, 14.1-14.5)
- **Machine Learning**
  - (18.1-18.4)
- Questions on any topic
- Pre-mid-term material if time and class interest
- Please review your quizzes, mid-term, & old tests
  - At least one question from a prior quiz or old CS-171 test will appear on the Final Exam (and all other tests)

# Review Bayesian Networks (Chapter 14.1-5)

- **You will be expected to know:**
- **Basic concepts and vocabulary of Bayesian networks.**
  - Nodes represent random variables.
  - Directed arcs represent (informally) direct influences.
  - Conditional probability tables,  $P(X_i \mid \text{Parents}(X_i))$ .
- **Given a Bayesian network:**
  - Write down the full joint distribution it represents.
  - Inference by Variable Elimination
- **Given a full joint distribution in factored form:**
  - Draw the Bayesian network that represents it.
- **Given a variable ordering and background assertions of conditional independence among the variables:**
  - Write down the factored form of the full joint distribution, as simplified by the conditional independence assertions.

# Bayesian Networks

---

- Represent dependence/independence via a directed graph
  - Nodes = random variables
  - Edges = direct dependence
- Structure of the graph  $\Leftrightarrow$  Conditional independence

- Recall the chain rule of repeated conditioning:

$$P(X_1, X_2, X_3, \dots, X_N) = P(X_1 | X_2, X_3, \dots, X_N) P(X_2 | X_3, \dots, X_N) \cdots P(X_N)$$

$$P(X_1, X_2, X_3, \dots, X_N) = \prod_{i=1}^n P(X_i | \text{parents}(X_i))$$

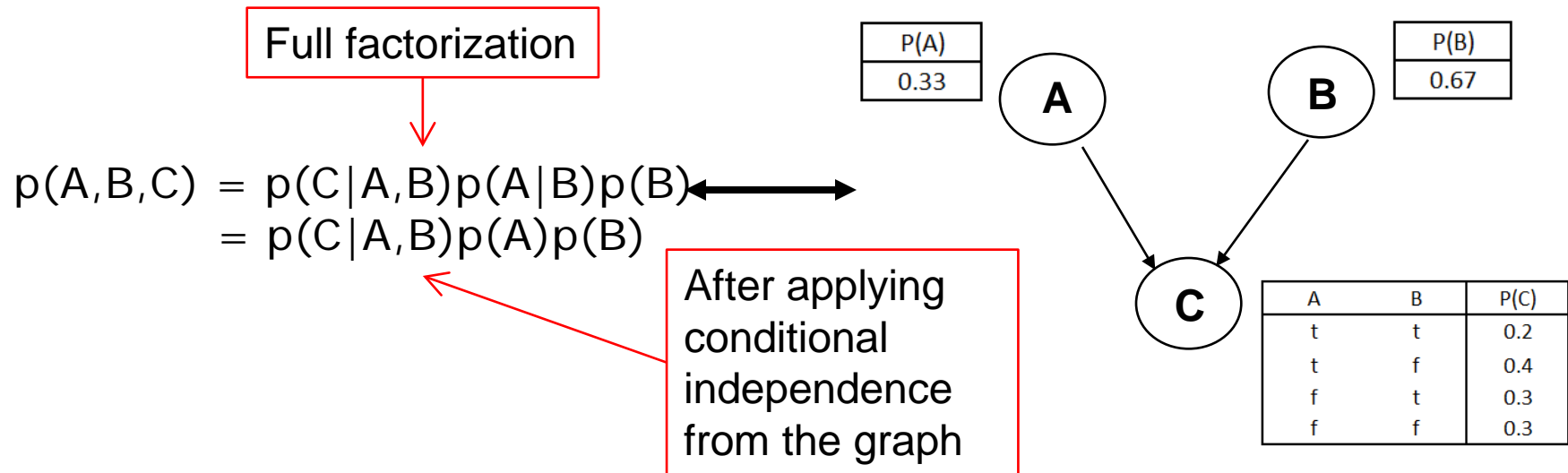
The full joint distribution

The graph-structured approximation

- Requires that graph is acyclic (no directed cycles)
- 2 components to a Bayesian network
  - The graph structure (conditional independence assumptions)
  - The numerical probabilities (of each variable given its parents)

# Bayesian Network

- A Bayesian network specifies a joint distribution in a structured form:



- Dependence/independence represented via a directed graph:
  - Node = random variable
  - Directed Edge = conditional dependence
  - Absence of Edge = conditional independence
- Allows concise view of joint distribution relationships:
  - Graph nodes and edges show conditional relationships between variables.
  - Tables provide probability data.

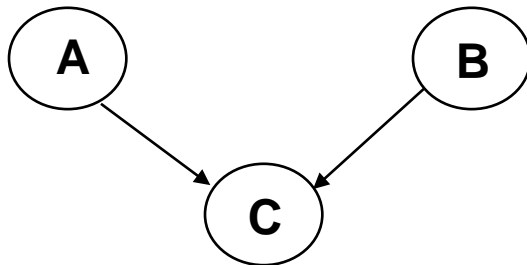
# Examples of 3-way Bayesian Networks

Independent Causes

A Earthquake

B Burglary

C Alarm



**Independent Causes:**

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

**“Explaining away” effect:**

**Given C, observing A makes B less likely  
e.g., earthquake/burglary/alarm example**

**A and B are (marginally) independent  
but become dependent once C is known**

**You heard alarm, and observe Earthquake  
.... It explains away burglary**

Nodes: Random Variables

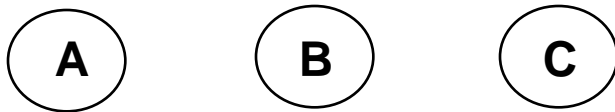
A, B, C

Edges:  $P(X_i | \text{Parents}) \rightarrow$  Directed edge from parent nodes to  $X_i$

$A \rightarrow C$

$B \rightarrow C$

# Examples of 3-way Bayesian Networks



**Marginal Independence:**  
 $p(A,B,C) = p(A) p(B) p(C)$

Nodes: Random Variables

A, B, C

Edges:  $P(X_i | \text{Parents}) \rightarrow$  Directed edge from parent nodes to  $X_i$

No Edge!

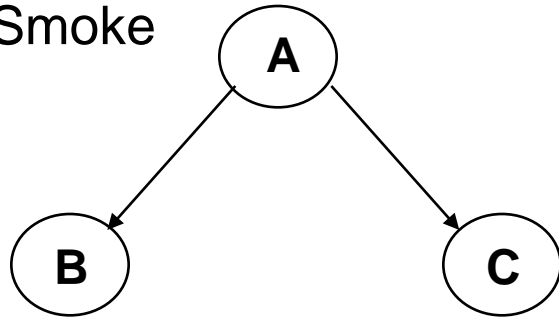
# Extended example of 3-way Bayesian Networks

Common Cause

A : Fire

B: Heat

C: Smoke



**Conditionally independent effects:**

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

**B and C are conditionally independent  
Given A**

**“Where there’s Smoke, there’s Fire.”**

**If we see Smoke, we can infer Fire.**

**If we see Smoke, observing Heat tells  
us very little additional information.**

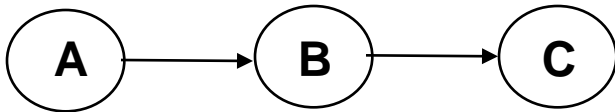
# Examples of 3-way Bayesian Networks

Markov Dependence

A Rain on Mon

B Ran on Tue

C Rain on Wed



**Markov dependence:**

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

**A affects B and B affects C**

**Given B, A and C are independent**

**e.g.**

**If it rains today, it will rain tomorrow with 90%**

**On Wed morning...**

**If you know it rained yesterday,**

**it doesn't matter whether it rained on Mon**

Nodes: Random Variables

A, B, C

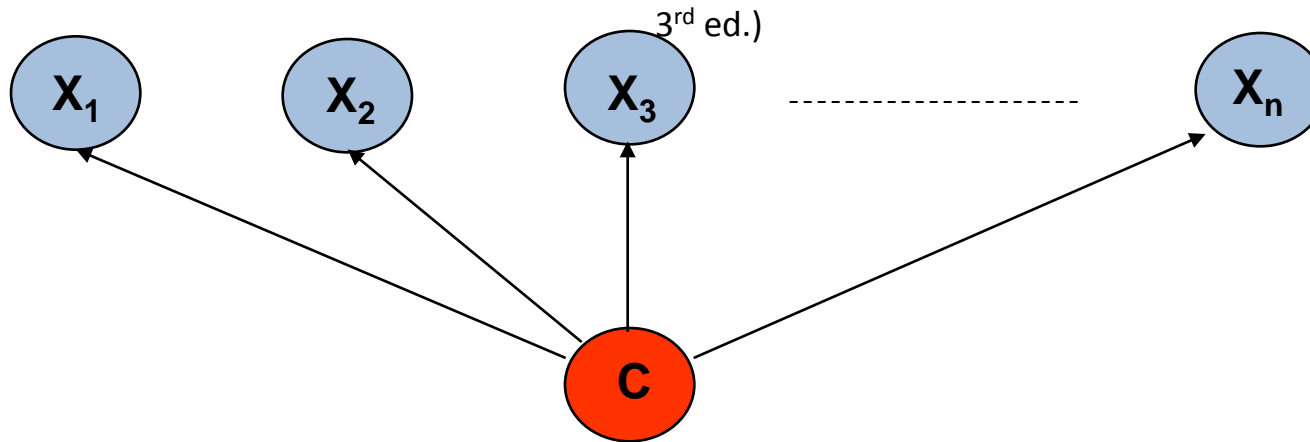
Edges:  $P(X_i | \text{Parents}) \rightarrow$  Directed edge from parent nodes to  $X_i$

$A \rightarrow B$

$B \rightarrow C$

# Naïve Bayes Model

(section 20.2.2 R&N



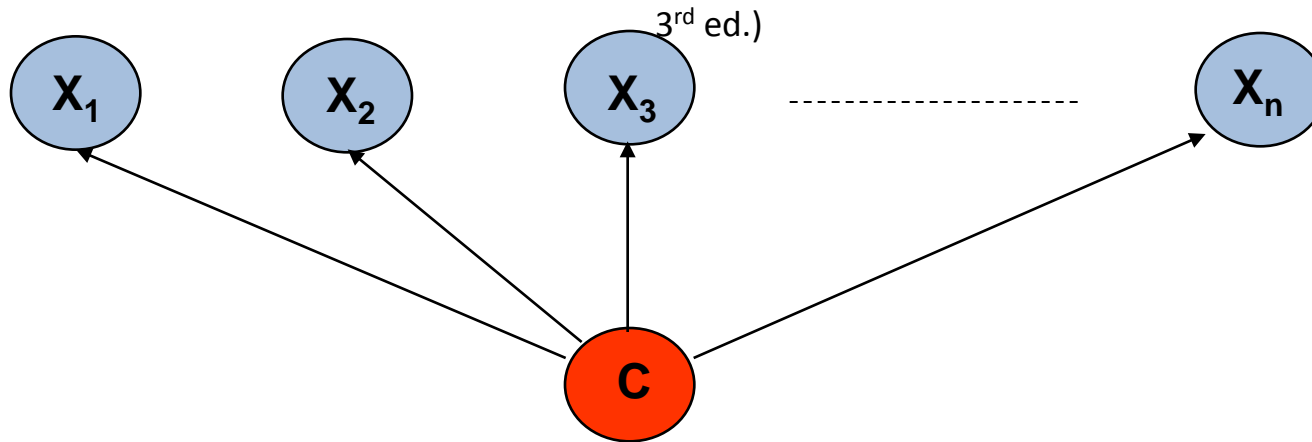
**Basic Idea:** We want to estimate  $P(C \mid X_1, \dots, X_n)$ , but it's hard to think about computing the probability of a class from input attributes of an example.

**Solution:** Use Bayes' Rule to turn  $P(C \mid X_1, \dots, X_n)$  into a proportionally equivalent expression that involves only  $P(C)$  and  $P(X_1, \dots, X_n \mid C)$ . Then assume that feature values are conditionally independent given class, which allows us to turn  $P(X_1, \dots, X_n \mid C)$  into  $\prod_i P(X_i \mid C)$ .

We estimate  $P(C)$  easily from the frequency with which each class appears within our training data, and we estimate  $P(X_i \mid C)$  easily from the frequency with which each  $X_i$  appears in each class  $C$  within our training data.

# Naïve Bayes Model

(section 20.2.2 R&N



**Bayes Rule:**  $P(C | X_1, \dots, X_n)$  is proportional to  $P(C) \prod_i P(X_i | C)$   
[note: denominator  $P(X_1, \dots, X_n)$  is constant for all classes, may be ignored.]

Features  $X_i$  are conditionally independent given the class variable  $C$

- choose the class value  $c_i$  with the highest  $P(c_i | x_1, \dots, x_n)$
- simple to implement, often works very well
- e.g., spam email classification:  $X$ 's = counts of words in emails

Conditional probabilities  $P(X_i | C)$  can easily be estimated from labeled data

- Problem: Need to avoid zeroes, e.g., from limited training data
- Solutions: Pseudo-counts, beta[a,b] distribution, etc.

# Naïve Bayes Model (2)

$$P(C | X_1, \dots, X_n) = \alpha P(C) \prod_i P(X_i | C)$$

Probabilities  $P(C)$  and  $P(X_i | C)$  can easily be estimated from labeled data

$$P(C = c_j) \approx \#(\text{Examples with class label } C = c_j) / \#(\text{Examples})$$

$$\begin{aligned} P(X_i = x_{ik} | C = c_j) \\ \approx \#(\text{Examples with attribute value } X_i = x_{ik} \text{ and class label } C = c_j) \\ / \#(\text{Examples with class label } C = c_j) \end{aligned}$$

Usually easiest to work with logs

$$\begin{aligned} \log [ P(C | X_1, \dots, X_n) ] \\ = \log \alpha + \log P(C) + \sum \log P(X_i | C) \end{aligned}$$

**DANGER:** What if ZERO examples with value  $X_i = x_{ik}$  and class label  $C = c_j$  ?  
An unseen example with value  $X_i = x_{ik}$  will NEVER predict class label  $C = c_j$  !

Practical solutions: Pseudocounts, e.g., add 1 to every  $\#()$  , etc.  
Theoretical solutions: Bayesian inference, beta distribution, etc.

## Bigger Example

---

- Consider the following 5 binary variables:
  - B = a burglary occurs at your house
  - E = an earthquake occurs at your house
  - A = the alarm goes off
  - J = John calls to report the alarm
  - M = Mary calls to report the alarm
- Sample Query: What is  $P(B|M, J)$  ?
- Using full joint distribution to answer this question requires
  - $2^5 - 1 = 31$  parameters
- Can we use prior domain knowledge to come up with a Bayesian network that requires fewer probabilities?

# Constructing a Bayesian Network: Step 1

- Order the variables in terms of influence (may be a partial order)

e.g.,  $\{E, B\} \rightarrow \{A\} \rightarrow \{J, M\}$

- $P(J, M, A, E, B) = P(J, M \mid A, E, B) P(A \mid E, B) P(E, B)$

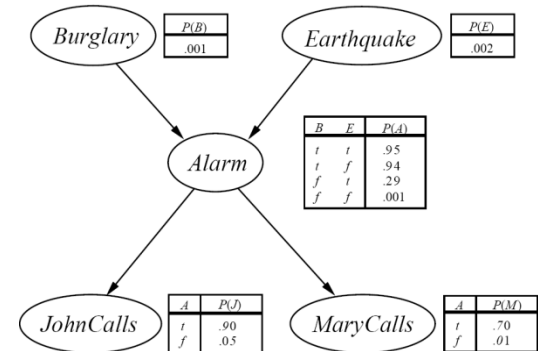
$$\approx P(J, M \mid A) \quad P(A \mid E, B) P(E) P(B)$$

$$\approx P(J \mid A) P(M \mid A) P(A \mid E, B) P(E) P(B)$$

These conditional independence assumptions are reflected in the graph structure of the Bayesian network

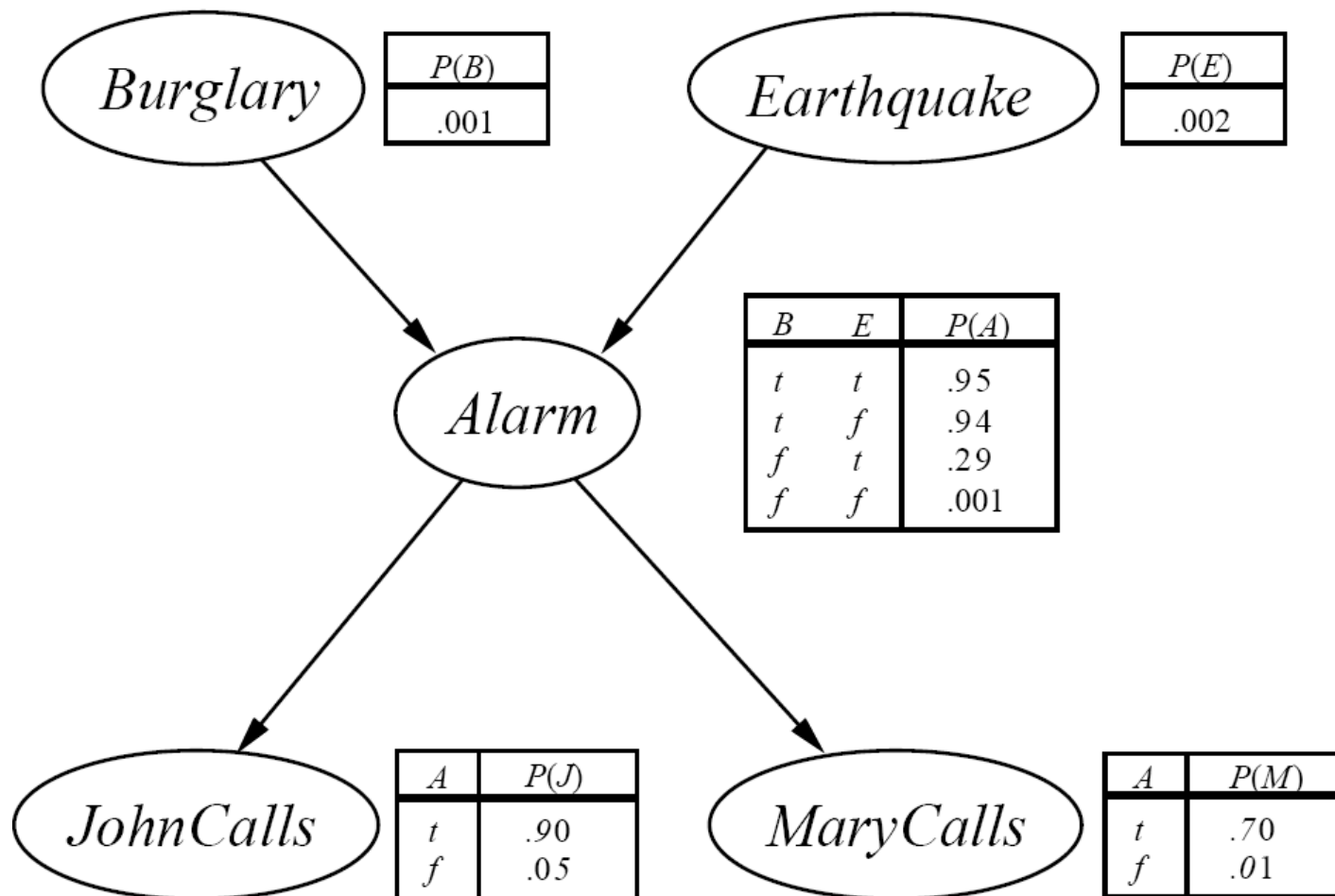
# Constructing this Bayesian Network: Step 2

- $P(J, M, A, E, B) =$   
 $P(J \mid A) P(M \mid A) P(A \mid E, B) P(E) P(B)$

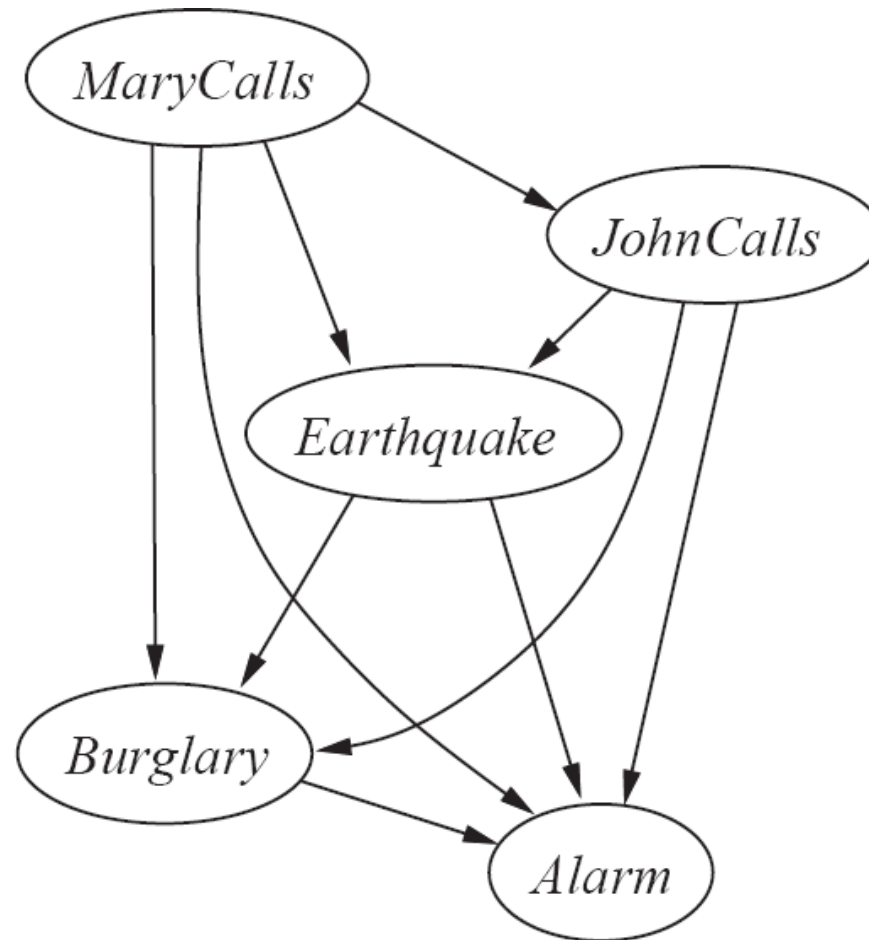


- There are 3 conditional probability tables (CPDs) to be determined:  
 $P(J \mid A)$ ,  $P(M \mid A)$ ,  $P(A \mid E, B)$ 
  - Requiring  $2 + 2 + 4 = 8$  probabilities
- And 2 marginal probabilities  $P(E)$ ,  $P(B)$  -> 2 more probabilities
- Where do these probabilities come from?
  - Expert knowledge
  - From data (relative frequency estimates)
  - Or a combination of both - see discussion in Section 20.1 and 20.2 (optional)

## The Resulting Bayesian Network



## The Bayesian Network from a different Variable Ordering

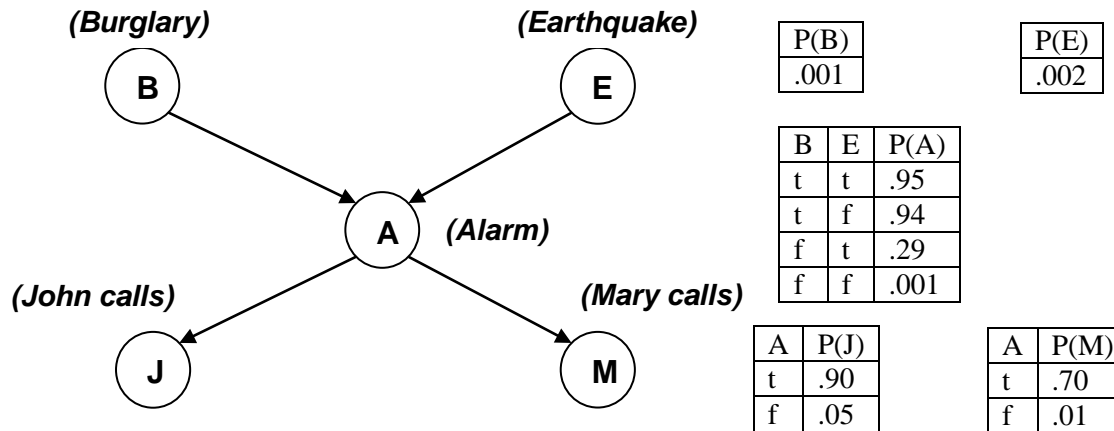


(b)

# Computing Probabilities from a Bayesian Network

Shown below is the Bayesian network for the Burglar Alarm problem, i.e.,

$$P(J,M,A,B,E) = P(J | A) P(M | A) P(A | B, E) P(B) P(E).$$



Suppose we wish to compute  $P( J=f \wedge M=t \wedge A=t \wedge B=t \wedge E=f )$ :

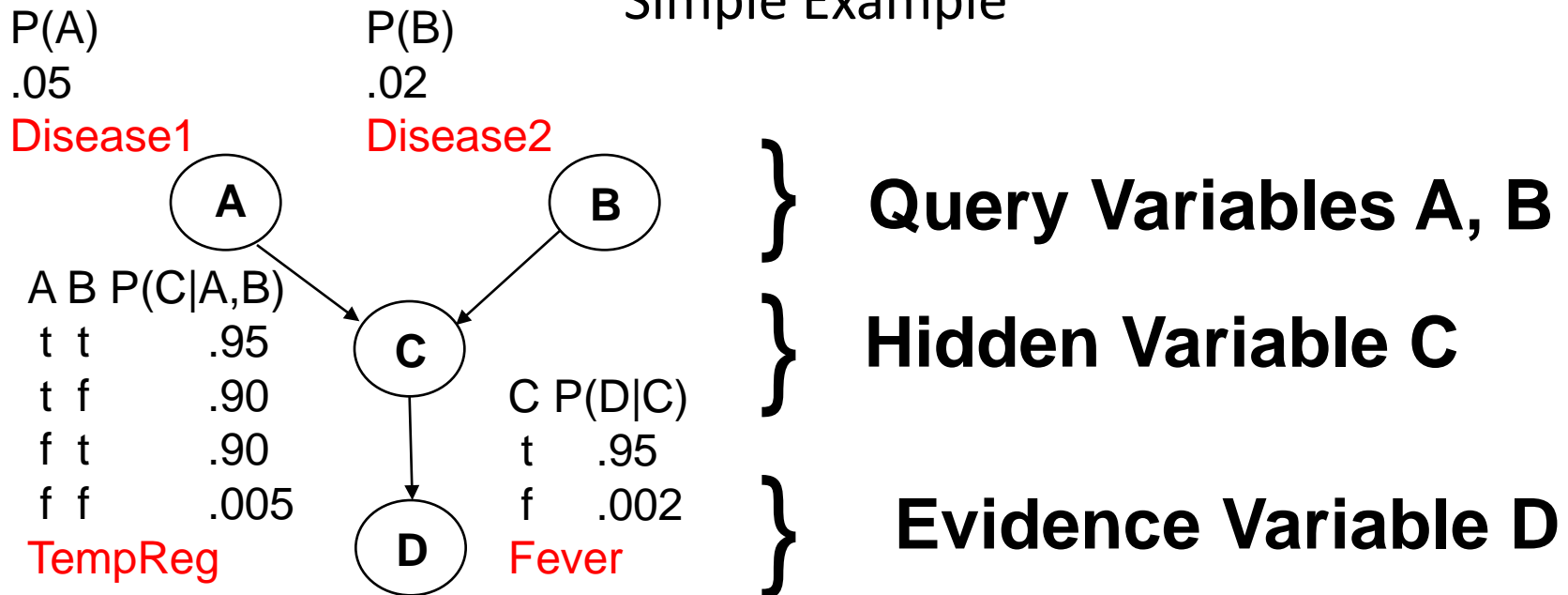
$$\begin{aligned}
 &P( J=f \wedge M=t \wedge A=t \wedge B=t \wedge E=f ) \\
 &= P( J=f | A=t ) * P( M=t | A=t ) * P( A=t | B=t \wedge E=f ) * P( B=t ) * P( E=f ) \\
 &= .10 * .70 * .94 * .001 * .998
 \end{aligned}$$

Note:  $P( E=f ) = [ 1 - P( E=t ) ] = [ 1 - .002 ] = .998$

$P( J=f | A=t ) = [ 1 - P( J=t | A=t ) ] = .10$

# Inference in Bayesian Networks

## Simple Example



(A=True, B=False | D=True) : Probability of getting Disease1 when we observe Fever

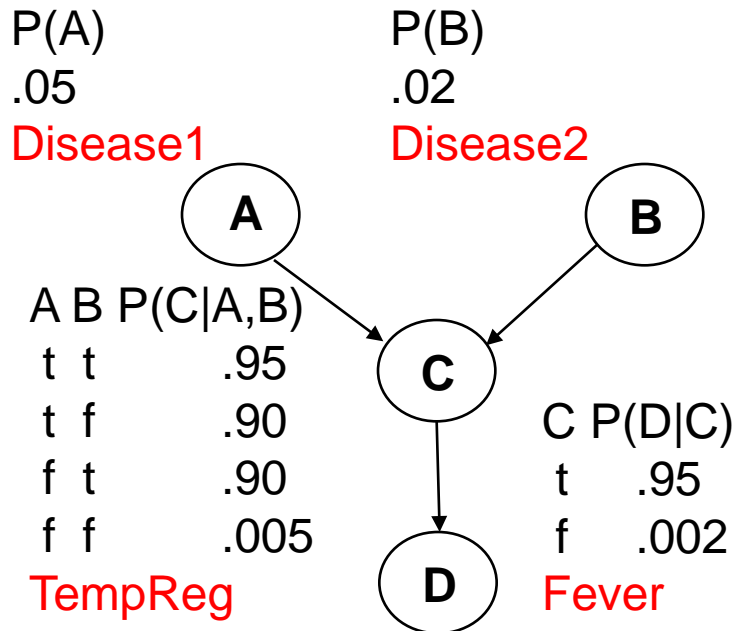
**Note: Not an anatomically correct model of how diseases cause fever!**

Suppose that two different diseases influence some imaginary internal body temperature regulator, which in turn influences whether fever is present.

# Inference in Bayesian Networks

- $\mathbf{X} = \{ X1, X2, \dots, Xk \}$  = **query variables** of interest
- $\mathbf{E} = \{ E1, \dots, El \}$  = **evidence variables** that are observed
- $\mathbf{Y} = \{ Y1, \dots, Ym \}$  = **hidden variables** (nonevidence, nonquery)
  
- **What is the posterior distribution of  $\mathbf{X}$ , given  $\mathbf{E}$ ?**
  - $P(\mathbf{X} \mid \mathbf{e}) = \alpha \sum_{\mathbf{y}} P(\mathbf{X}, \mathbf{y}, \mathbf{e})$   
Normalizing constant  $\alpha = \sum_{\mathbf{x}} \sum_{\mathbf{y}} P(\mathbf{X}, \mathbf{y}, \mathbf{e})$
  
- **What is the most likely assignment of values to  $\mathbf{X}$ , given  $\mathbf{E}$ ?**
  - $\operatorname{argmax}_{\mathbf{x}} P(\mathbf{x} \mid \mathbf{e}) = \operatorname{argmax}_{\mathbf{x}} \sum_{\mathbf{y}} P(\mathbf{x}, \mathbf{y}, \mathbf{e})$

# Inference by Variable Elimination



What is the posterior conditional distribution of our query variables, given that fever was observed?

$$\begin{aligned}
 P(A,B|d) &= \alpha \sum_c P(A,B,c,d) \\
 &= \alpha \sum_c P(A)P(B)P(c|A,B)P(d|c) \\
 &= \alpha P(A)P(B) \sum_c P(c|A,B)P(d|c)
 \end{aligned}$$

$$\begin{aligned}
 P(a,b|d) &= \alpha P(a)P(b) \sum_c P(c|a,b)P(d|c) = \alpha P(a)P(b) \{ P(c|a,b)P(d|c) + P(\neg c|a,b)P(d|\neg c) \} \\
 &= \alpha .05 \times .02 \times \{ .95 \times .95 + .05 \times .002 \} \approx \alpha .000903 \approx .014
 \end{aligned}$$

$$\begin{aligned}
 P(\neg a,b|d) &= \alpha P(\neg a)P(b) \sum_c P(c|\neg a,b)P(d|c) = \alpha P(\neg a)P(b) \{ P(c|\neg a,b)P(d|c) + P(\neg c|\neg a,b)P(d|\neg c) \} \\
 &= \alpha .95 \times .02 \times \{ .90 \times .95 + .10 \times .002 \} \approx \alpha .0162 \approx .248
 \end{aligned}$$

$$\begin{aligned}
 P(a,\neg b|d) &= \alpha P(a)P(\neg b) \sum_c P(c|a,\neg b)P(d|c) = \alpha P(a)P(\neg b) \{ P(c|a,\neg b)P(d|c) + P(\neg c|a,\neg b)P(d|\neg c) \} \\
 &= \alpha .05 \times .98 \times \{ .90 \times .95 + .10 \times .002 \} \approx \alpha .0419 \approx .642
 \end{aligned}$$

$$\begin{aligned}
 P(\neg a,\neg b|d) &= \alpha P(\neg a)P(\neg b) \sum_c P(c|\neg a,\neg b)P(d|c) = \alpha P(\neg a)P(\neg b) \{ P(c|\neg a,\neg b)P(d|c) + P(\neg c|\neg a,\neg b)P(d|\neg c) \} \\
 &= \alpha .95 \times .98 \times \{ .005 \times .95 + .995 \times .002 \} \approx \alpha .00627 \approx .096
 \end{aligned}$$

$$\alpha \approx 1 / (.000903 + .0162 + .0419 + .00627) \approx 1 / .06527 \approx 15.32 \quad \text{[Note: } \alpha = \text{normalization constant, p. 493]}$$

# CS-171 Final Review

---

- **Propositional Logic**
  - (7.1-7.5)
- **First-Order Logic, Knowledge Representation**
  - (8.1-8.5, 9.1-9.2)
- **Probability & Bayesian Networks**
  - (13, 14.1-14.5)
- **Machine Learning**
  - (18.1-18.4)
- Questions on any topic
- Pre-mid-term material if time and class interest
- Please review your quizzes, mid-term, & old tests
  - At least one question from a prior quiz or old CS-171 test will appear on the Final Exam (and all other tests)

# The importance of a good representation

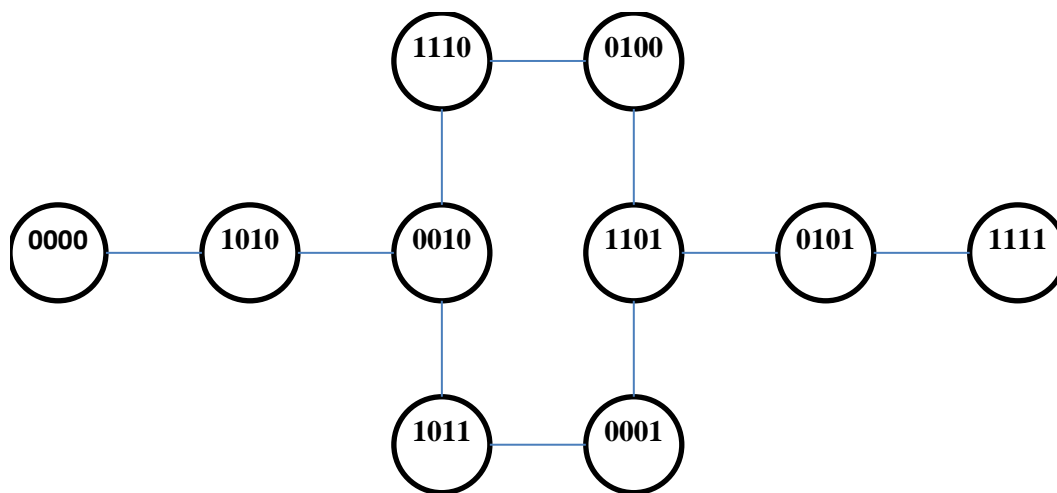
---

- **Properties of a good representation:**
- Reveals important features
- Hides irrelevant detail
- Exposes useful constraints
- Makes frequent operations easy-to-do
- Supports local inferences from local features
  - Called the “soda straw” principle or “locality” principle
  - Inference from features “through a soda straw”
- Rapidly or efficiently computable
  - It’s nice to be fast

## Reveals important features / Hides irrelevant detail

---

- “You can’t learn what you can’t represent.” --- G. Sussman
- **In search:** *A man is traveling to market with a fox, a goose, and a bag of oats. He comes to a river. The only way across the river is a boat that can hold the man and exactly one of the fox, goose or bag of oats. The fox will eat the goose if left alone with it, and the goose will eat the oats if left alone with it.*
- **A good representation makes this problem easy:**



# Terminology

---

- Attributes
  - Also known as features, variables, independent variables, covariates
- Target Variable
  - Also known as goal predicate, dependent variable, ...
- Classification
  - Also known as discrimination, supervised classification, ...
- Error function
  - Objective function, loss function, ...

# Inductive learning

---

- Let  $x$  represent the input vector of attributes
- Let  $f(x)$  represent the value of the target variable for  $x$ 
  - The implicit mapping from  $x$  to  $f(x)$  is unknown to us
  - We just have training data pairs,  $D = \{x, f(x)\}$  available
- We want to learn a mapping from  $x$  to  $f$ , i.e.,  
 $h(x; \theta)$  is “close” to  $f(x)$  for all training data points  $x$

$\theta$  are the parameters of our predictor  $h(..)$

- Examples:
  - $h(x; \theta) = \text{sign}(w_1x_1 + w_2x_2 + w_3)$
  - $h_k(x) = (x_1 \text{ OR } x_2) \text{ AND } (x_3 \text{ OR NOT}(x_4))$

# Empirical Error Functions

---

- Empirical error function:

$$E(h) = \sum_x \text{distance}[h(x; \theta) , f]$$

e.g., distance = squared error if h and f are real-valued (regression)

distance = delta-function if h and f are categorical (classification)

Sum is over all training pairs in the training data D

In learning, we get to choose

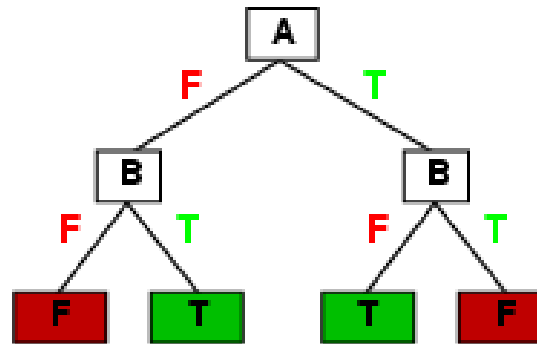
1. what class of functions  $h(..)$  that we want to learn
  - potentially a huge space! (“hypothesis space”)
2. what error function/distance to use
  - should be chosen to reflect real “loss” in problem
  - but often chosen for mathematical/algorithmic convenience

# Decision Tree Representations

---

- Decision trees are fully expressive
  - can represent any Boolean function
  - Every path in the tree could represent 1 row in the truth table
  - Yields an exponentially large tree
    - Truth table is of size  $2^d$ , where  $d$  is the number of attributes

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



# Pseudocode for Decision tree learning

---

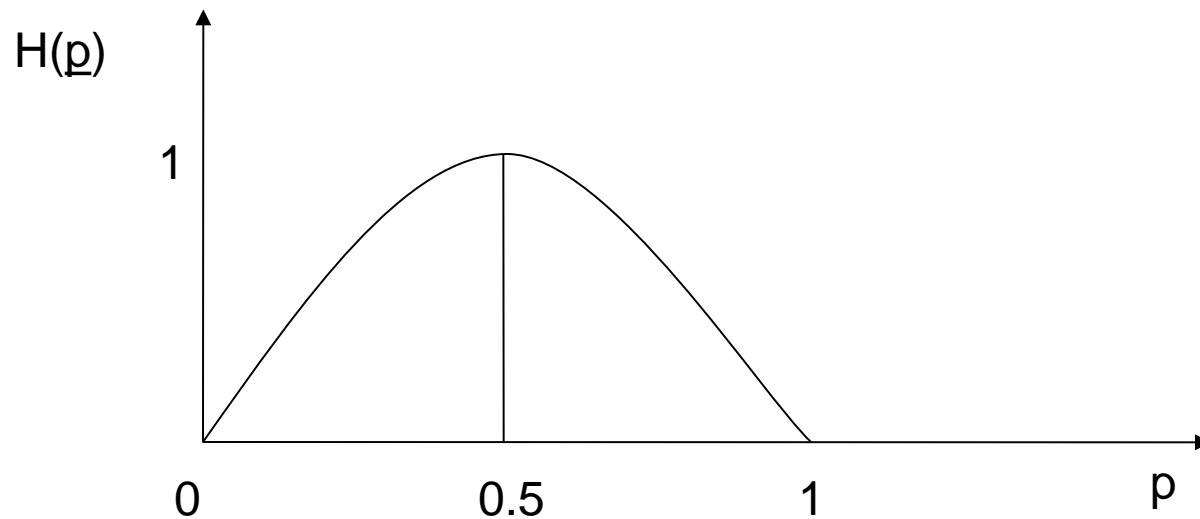
```
function DTL(examples, attributes, default) returns a decision tree
  if examples is empty then return default
  else if all examples have the same classification then return the classification
  else if attributes is empty then return MODE(examples)
  else
    best ← CHOOSE-ATTRIBUTE(attributes, examples)
    tree ← a new decision tree with root test best
    for each value  $v_i$  of best do
       $examples_i \leftarrow \{\text{elements of } examples \text{ with } best = v_i\}$ 
      subtree ← DTL(examplesi, attributes – best, MODE(examples))
      add a branch to tree with label  $v_i$  and subtree subtree
  return tree
```

## Entropy with only 2 outcomes

---

Consider 2 class problem:  $p$  = probability of class 1,  $1 - p$  = probability of class 2

In binary case,  $H(p) = -p \log p - (1-p) \log (1-p)$



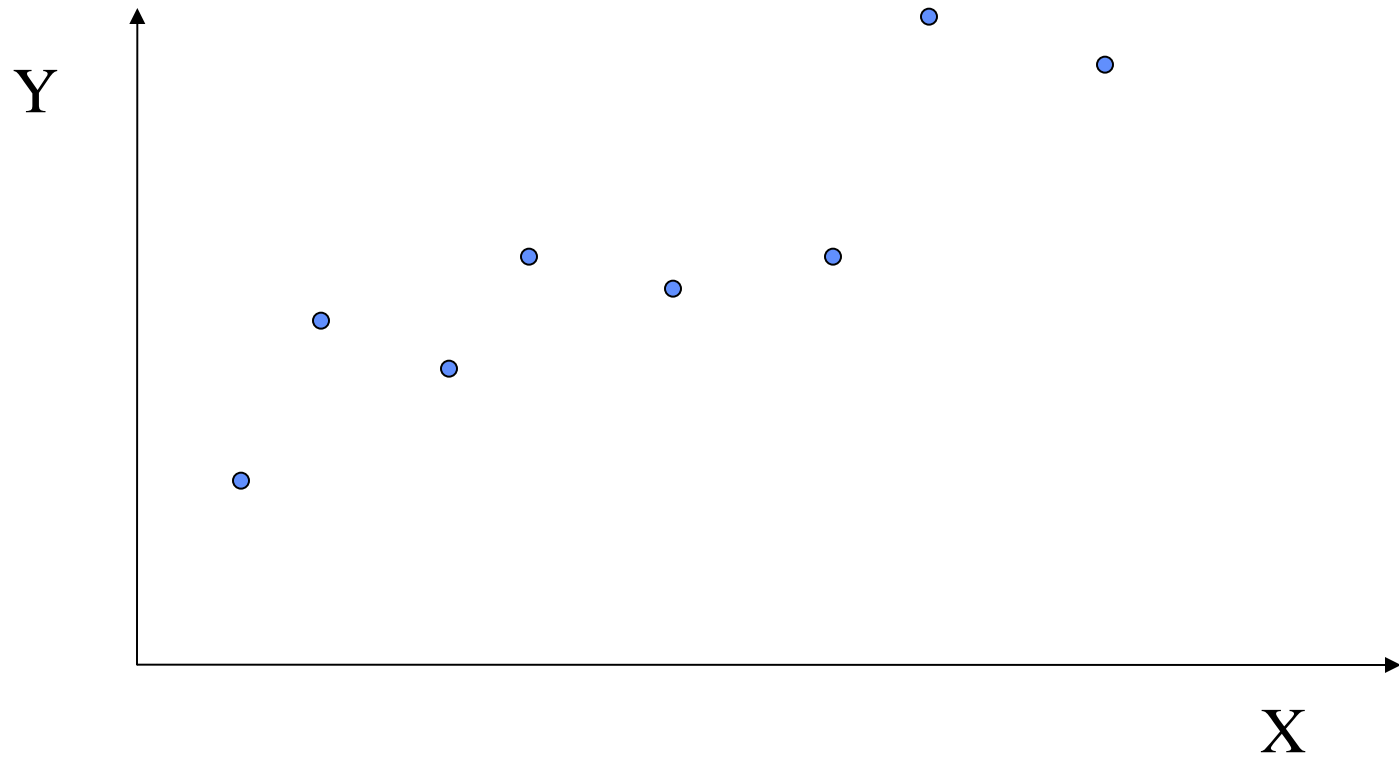
# Information Gain

---

- $H(p)$  = entropy of class distribution at a particular node
- $H(p \mid A)$  = conditional entropy = average entropy of conditional class distribution, after we have partitioned the data according to the values in  $A$
- $\text{Gain}(A) = H(p) - H(p \mid A)$
- Simple rule in decision tree learning
  - At each internal node, split on the node with the largest information gain (or equivalently, with smallest  $H(p \mid A)$ )
- Note that by definition, conditional entropy can't be greater than the entropy

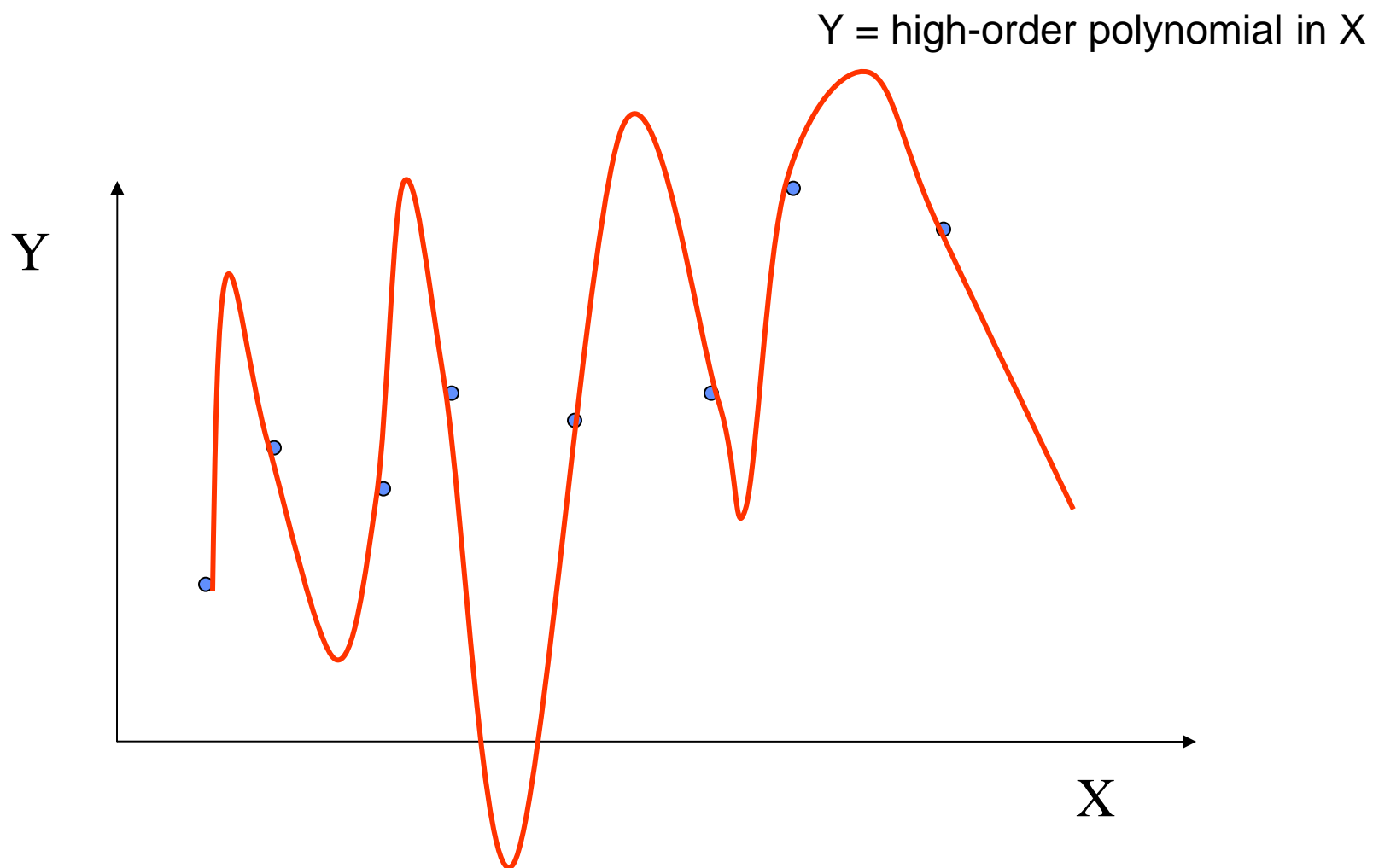
# Overfitting and Underfitting

---



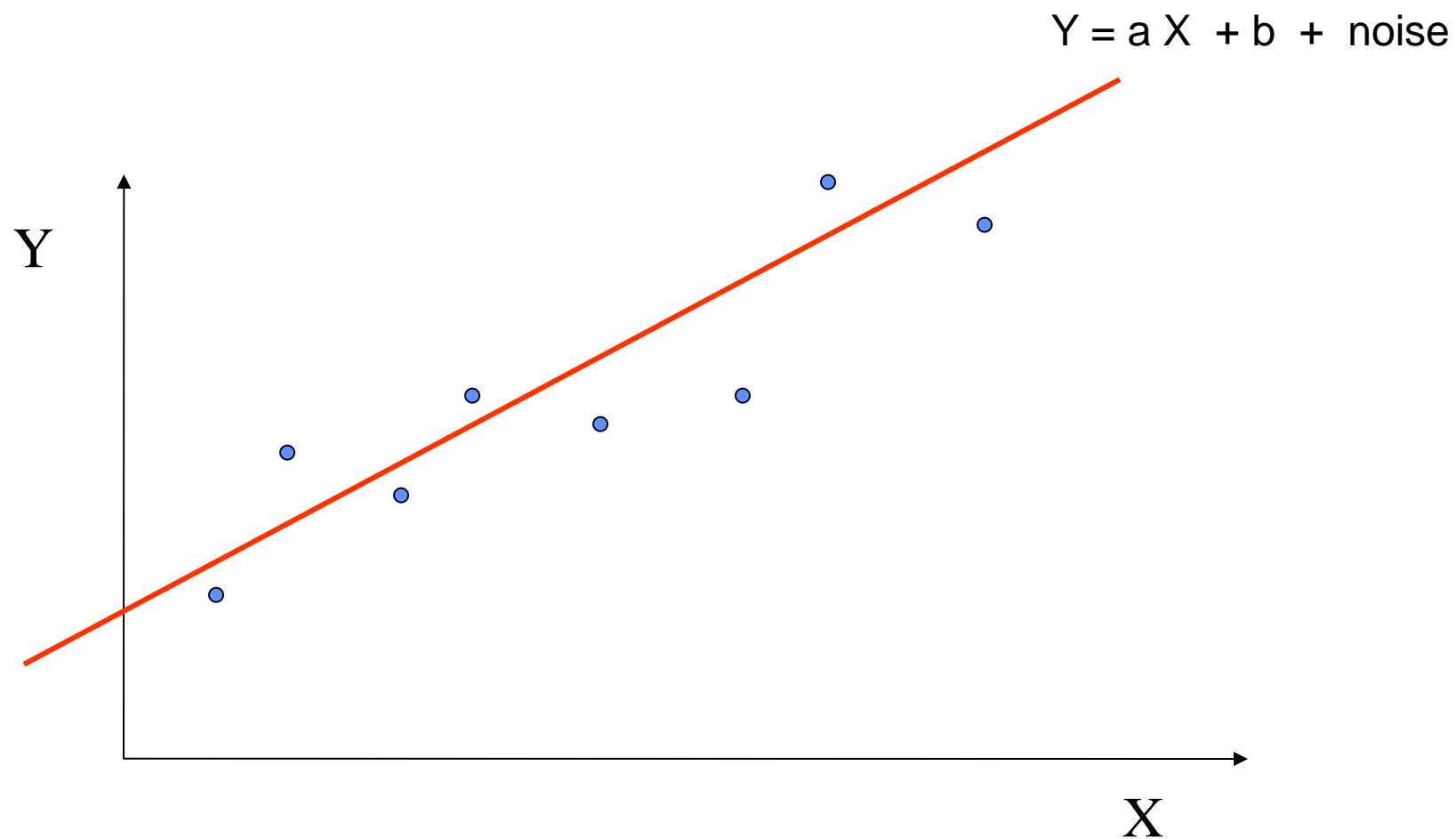
## A Complex Model

---



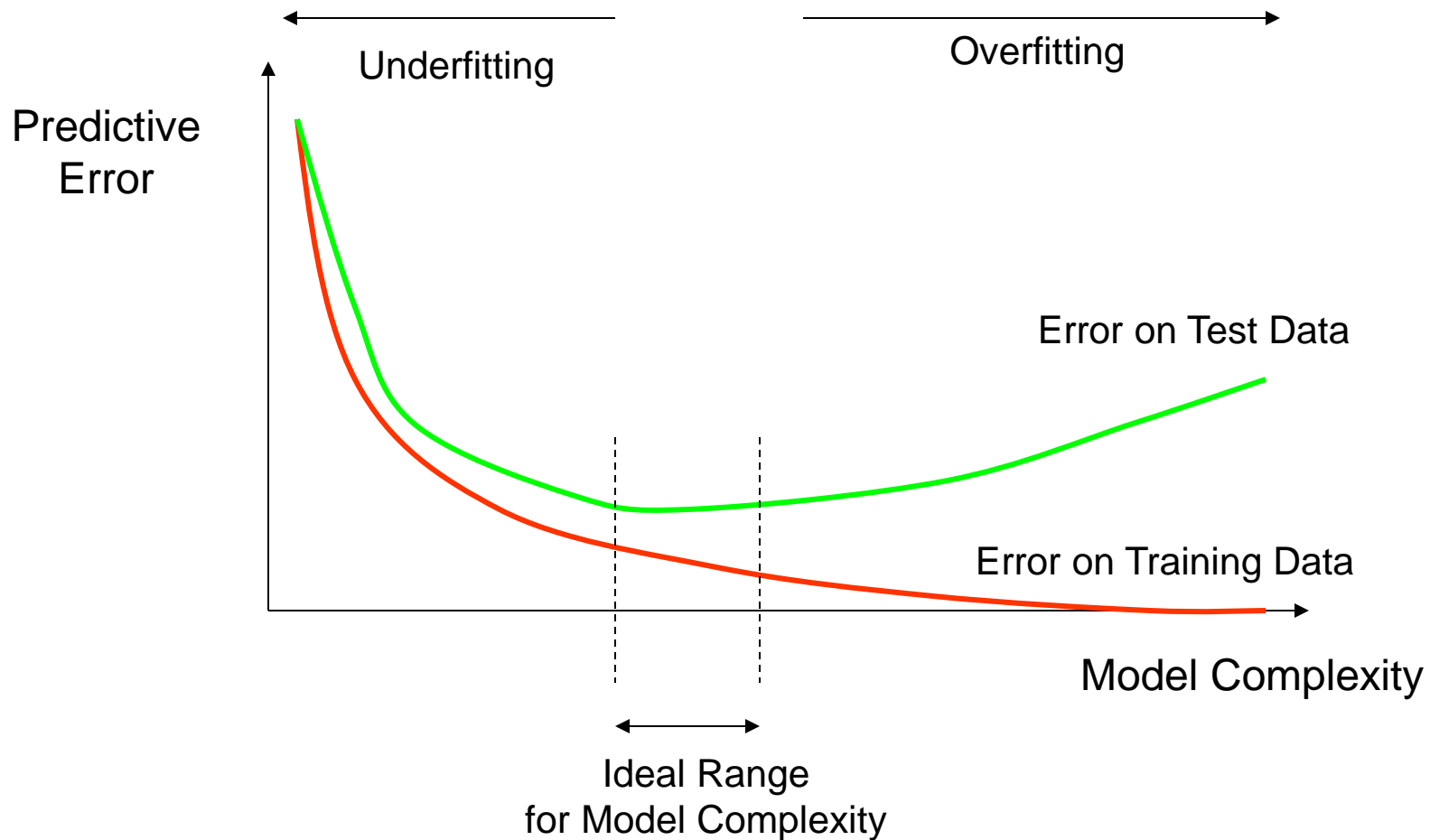
## A Much Simpler Model

---



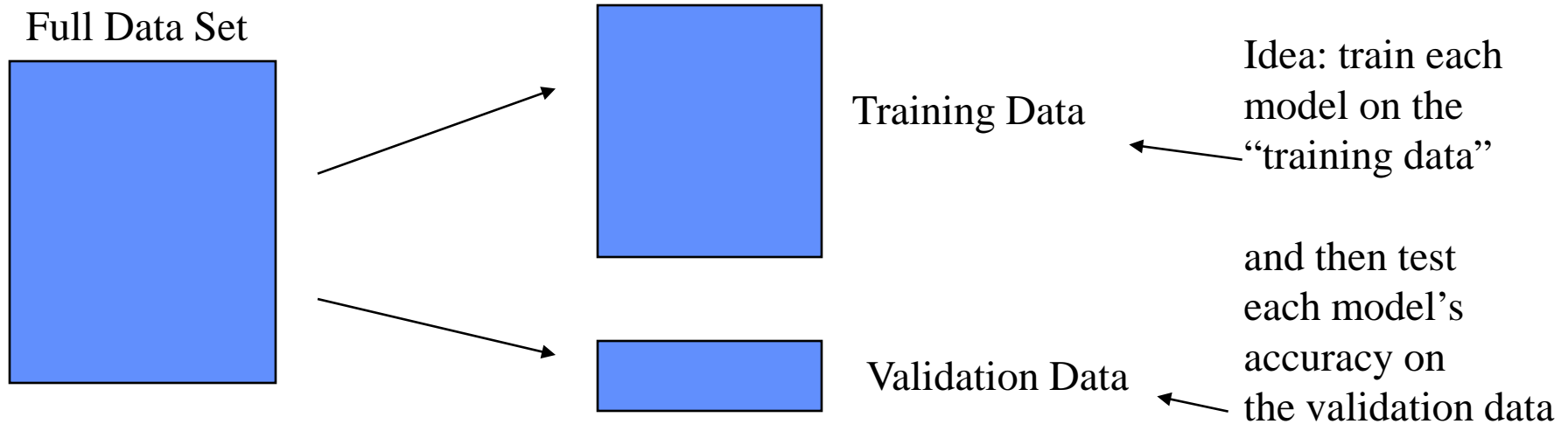
# How Overfitting affects Prediction

---



# Training and Validation Data

---



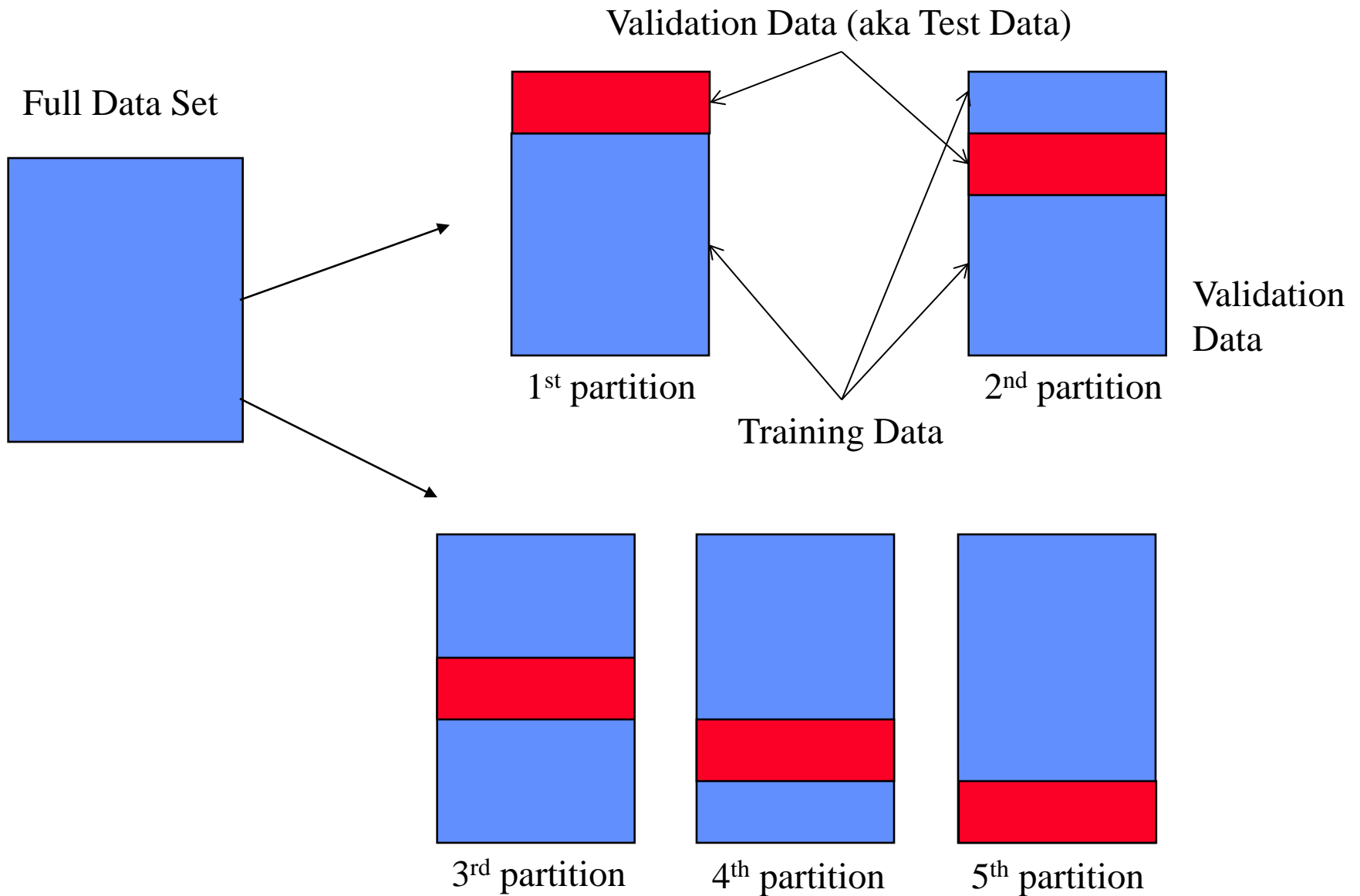
# The k-fold Cross-Validation Method

---

- Why just choose one particular 90/10 “split” of the data?
  - In principle we could do this multiple times
- “k-fold Cross-Validation” (e.g.,  $k=10$ )
  - randomly partition our full data set into  $k$  disjoint subsets (each roughly of size  $n/k$ ,  $n$  = total number of training data points)
    - for  $i = 1:10$  (here  $k = 10$ )
      - train on 90% of data,
      - $\text{Acc}(i)$  = accuracy on other 10%
    - end
    - $\text{Cross-Validation-Accuracy} = 1/k \sum_i \text{Acc}(i)$
  - choose the method with the highest cross-validation accuracy
  - common values for  $k$  are 5 and 10
  - Can also do “leave-one-out” where  $k = n$

# Disjoint Validation Data Sets

---



# CS-171 Final Review

---

- **Propositional Logic**
  - (7.1-7.5)
- **First-Order Logic, Knowledge Representation**
  - (8.1-8.5, 9.1-9.2)
- **Probability & Bayesian Networks**
  - (13, 14.1-14.5)
- **Machine Learning**
  - (18.1-18.4)
- Questions on any topic
- Pre-mid-term material if time and class interest
- Please review your quizzes, mid-term, & old tests
  - At least one question from a prior quiz or old CS-171 test will appear on the Final Exam (and all other tests)