

# Propositional Logic

# Big Ideas

- Logic is a great knowledge representation language for many AI problems
- **Propositional logic** is the simple foundation and fine for some AI problems
- **First order logic (FOL)** is much more expressive as a KR language and more commonly used in AI
- There are many variations: horn logic, higher order logic, three-valued logic, probabilistic logics, etc.

# Propositional logic

- **Logical constants:** true, false
- **Propositional symbols:** P, Q,... (**atomic sentences**)
- **Wrapping parentheses:** ( ... )
- Sentences are combined by **connectives:**
  - and [conjunction]
  - or [disjunction]
  - implies [implication / conditional]
  - is equivalent [biconditional]
  - not [negation]
- **Literal:** atomic sentence or negated atomic sentence  
P,  $\neg P$

# Examples of PL sentences

•  $(P \wedge Q) \rightarrow R$

“If it is hot and humid, then it is raining”

•  $Q \rightarrow P$

“If it is humid, then it is hot”

•  $Q$

“It is humid.”

• We're free to choose better symbols, btw:

$H_o$  = “It is hot”

$H_u$  = “It is humid”

$R$  = “It is raining”

# Propositional logic (PL)

- Simple language for showing key ideas and definitions
- User defines set of propositional symbols, like P and Q
- User defines **semantics** of each propositional symbol:
  - P means “It is hot”, Q means “It is humid”, etc.
- A sentence (well formed formula) is defined as follows:
  - A symbol is a sentence
  - If S is a sentence, then  $\neg S$  is a sentence
  - If S is a sentence, then  $(S)$  is a sentence
  - If S and T are sentences, then  $(S \wedge T)$ ,  $(S \vee T)$ ,  $(S \supset T)$ , and  $(S \leftrightarrow T)$  are sentences
  - A sentence results from a finite number of applications of the rules

# Some terms

- The meaning or **semantics** of a sentence determines its **interpretation**
- Given the truth values of all symbols in a sentence, it can be “evaluated” to determine its **truth value** (True or False)
- A **model** for a KB is a *possible world* – an assignment of truth values to propositional symbols that makes each sentence in the KB True

# Model for a KB

- Let the KB be  $[P \supset Q, R, Q \supset P]$
- What are the possible models? Consider all possible assignments of T|F to P, Q and R and check truth tables
  - **FFF: OK**
  - **FFT: OK**
  - FTF: NO
  - FTT: NO
  - **TFF: OK**
  - **TFT: OK**
  - TTF: NO
  - **TTT: OK**
- If KB is  $[P \supset Q, R, Q \supset P, Q]$ , then the only model is TTT

P: it's hot  
Q: it's humid  
R: it's raining

# More terms

- A **valid sentence** or **tautology** is a sentence that is True under all interpretations, no matter what the world is actually like or what the semantics is. Example: “It’s raining or it’s not raining”
- An **inconsistent sentence** or **contradiction** is a sentence that is False under all interpretations. The world is never like what it describes, as in “It’s raining and it’s not raining.”
- **P entails Q**, written  $P \models Q$ , means that whenever P is True, so is Q. In other words, all models of P are also models of Q.



# Truth tables

- Truth tables are used to define logical connectives
- and to determine when a complex sentence is true given the values of the symbols in it

*Truth tables for the five logical connectives*

$P$	$Q$	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
<i>False</i>	<i>False</i>	<i>True</i>	<i>False</i>	<i>False</i>	<i>True</i>	<i>True</i>
<i>False</i>	<i>True</i>	<i>True</i>	<i>False</i>	<i>True</i>	<i>True</i>	<i>False</i>
<i>True</i>	<i>False</i>	<i>False</i>	<i>False</i>	<i>True</i>	<i>False</i>	<i>False</i>
<i>True</i>	<i>True</i>	<i>False</i>	<i>True</i>	<i>True</i>	<i>True</i>	<i>True</i>

*Example of a truth table used for a complex sentence*

$P$	$H$	$P \vee H$	$(P \vee H) \wedge \neg H$	$((P \vee H) \wedge \neg H) \Rightarrow P$
<i>False</i>	<i>False</i>	<i>False</i>	<i>False</i>	<i>True</i>
<i>False</i>	<i>True</i>	<i>True</i>	<i>False</i>	<i>True</i>
<i>True</i>	<i>False</i>	<i>True</i>	<i>True</i>	<i>True</i>
<i>True</i>	<i>True</i>	<i>True</i>	<i>False</i>	<i>True</i>

# On the implies connective: $P \implies Q$

- Note that  $\implies$  is a logical connective
- So  $P \implies Q$  is a logical sentence and has a truth value, i.e., is either true or false
- If we add this sentence to the KB, it can be used by an inference rule, *Modes Ponens*, to derive/infer/prove  $Q$  if  $P$  is also in the KB
- Given a KB where  $P=\text{True}$  and  $Q=\text{True}$ , we can also derive/infer/prove that  $P \implies Q$  is True

# P Q

- When is  $P \quad Q$  true? Check all that apply
  - $P=Q=true$
  - $P=Q=false$
  - $P=true, Q=false$
  - $P=false, Q=true$

# P Q

- When is  $P \rightarrow Q$  true? Check all that apply
  - $P=Q=true$
  - $P=Q=false$
  - $P=true, Q=false$
  - $P=false, Q=true$
- We can get this from the truth table for
- Note: in FOL it's much harder to prove that a conditional true.
  - Consider proving  $\text{prime}(x) \rightarrow \text{odd}(x)$

# Inference rules

- **Logical inference** creates new sentences that logically follow from a set of sentences (KB)
- An inference rule is **sound** if every sentence  $X$  it produces when operating on a KB logically follows from the KB
  - i.e., inference rule creates no contradictions
- An inference rule is **complete** if it can produce every expression that logically follows from (is entailed by) the KB.
  - Note analogy to complete search algorithms

# Sound rules of inference

- Here are some examples of sound rules of inference
- Each can be shown to be sound using a truth table

<u>RULE</u>	<u>PREMISE</u>	<u>CONCLUSION</u>
Modus Ponens	$A, A$	$B$
And Introduction	$A, B$	$A \wedge B$
And Elimination	$A \wedge B$	$A$
Double Negation	$\neg\neg A$	$A$
Unit Resolution	$A \vee B, B$	$A$
<b>Resolution</b>	<b><math>A \vee B, B \vee C</math></b>	<b><math>A \vee C</math></b>

# Soundness of modus ponens

<b>A</b>	<b>B</b>	<b><math>A \rightarrow B</math></b>	<b>OK?</b>
True	True	True	
True	False	False	
False	True	True	
False	False	True	

# Resolution

- **Resolution** is a valid inference rule producing a new clause implied by two clauses containing *complementary literals*
  - A literal is an atomic symbol or its negation, i.e.,  $P$ ,  $\sim P$
- Amazingly, this is the only inference rule you need to build a sound and complete theorem prover
  - Based on proof by contradiction and usually called resolution refutation
- The resolution rule was discovered by Alan Robinson (CS, U. of Syracuse) in the mid 60s



# Resolution

- A KB is actually a set of sentences all of which are true, i.e., a conjunction of sentences.
- To use resolution, put KB into *conjunctive normal form* (CNF), where each sentence written as a disjunction of (one or more) literals

## Example

- KB:  $[P \vee Q, Q \vee R \vee S]$
- KB in CNF:  $[\sim P \vee Q, \sim Q \vee R, \sim Q \vee S]$
- Resolve KB(1) and KB(2) producing:  $\sim P \vee R$  (*i.e.*,  $P \vee R$ )
- Resolve KB(1) and KB(3) producing:  $\sim P \vee S$  (*i.e.*,  $P \vee S$ )
- New KB:  $[\sim P \vee Q, \sim Q \vee \sim R \vee \sim S, \sim P \vee R, \sim P \vee S]$

## Tautologies

$$(A \vee B) \leftrightarrow (\sim A \vee B)$$

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

# Soundness of the resolution inference rule

$\alpha$	$\beta$	$\gamma$	$\alpha \vee \beta$	$\neg\beta \vee \gamma$	$\alpha \vee \gamma$
<i>False</i>	<i>False</i>	<i>False</i>	<i>False</i>	<i>True</i>	<i>False</i>
<i>False</i>	<i>False</i>	<i>True</i>	<i>False</i>	<i>True</i>	<i>True</i>
<i>False</i>	<i>True</i>	<i>False</i>	<i>True</i>	<i>False</i>	<i>False</i>
<u><i>False</i></u>	<u><i>True</i></u>	<u><i>True</i></u>	<u><i>True</i></u>	<u><i>True</i></u>	<u><i>True</i></u>
<u><i>True</i></u>	<u><i>False</i></u>	<u><i>False</i></u>	<u><i>True</i></u>	<u><i>True</i></u>	<u><i>True</i></u>
<u><i>True</i></u>	<u><i>False</i></u>	<u><i>True</i></u>	<u><i>True</i></u>	<u><i>True</i></u>	<u><i>True</i></u>
<i>True</i>	<i>True</i>	<i>False</i>	<i>True</i>	<i>False</i>	<i>True</i>
<u><i>True</i></u>	<u><i>True</i></u>	<u><i>True</i></u>	<u><i>True</i></u>	<u><i>True</i></u>	<u><i>True</i></u>

From the rightmost three columns of this truth table, we can see that

$$(\alpha \quad \beta) \quad (\beta \quad \gamma) \leftrightarrow (\alpha \quad \gamma)$$

is valid (i.e., always true regardless of the truth values assigned to  $\alpha$ ,  $\beta$  and  $\gamma$ )



# Horn sentences

- A **Horn sentence** or **Horn clause** has the form:

$P_1 \quad P_2 \quad P_3 \quad \dots \quad P_n \quad \rightarrow \quad Q_m$  where  $n \geq 0, m \in \{0, 1\}$

- Note: a conjunction of 0 or more symbols to left of  $\rightarrow$  and 0-1 symbols to right
- Special cases:
  - $n=0, m=1$ :  $P$  (assert  $P$  is true)
  - $n>0, m=0$ :  $P \quad Q$  (constraint: both  $P$  and  $Q$  can't be true)
  - $n=0, m=0$ : (well, there is nothing there!)
- Put in CNF: each sentence is a disjunction of literals with at most one non-negative literal

$P_1 \quad P_2 \quad P_3 \quad \dots \quad P_n \quad Q$

$$(P \quad Q) = ( \quad P \quad Q )$$

# Significance of Horn logic

- We can also have horn sentences in FOL
- Reasoning with horn clauses is much simpler
  - Satisfiability of a propositional KB (i.e., finding values for a symbols that will make it true) is NP complete
  - Restricting KB to horn sentences, satisfiability is in P
- For this reason, FOL Horn sentences are the basis for Prolog and Datalog
- What Horn sentences give up are handling, in a general way, (1) negation and (2) disjunctions

# Entailment and derivation

- **Entailment:  $KB \models Q$** 
  - $Q$  is entailed by  $KB$  (set sentences) iff there is no logically possible world where  $Q$  is false while all the sentences in  $KB$  are true
  - Or, stated positively,  $Q$  is entailed by  $KB$  iff the conclusion is true in every logically possible world in which all the premises in  $KB$  are true
- **Derivation:  $KB \vdash Q$** 
  - We can derive  $Q$  from  $KB$  if there's a proof consisting of a sequence of valid inference steps starting from the premises in  $KB$  and resulting in  $Q$

# Two important properties for inference

**Soundness: If  $KB \vdash Q$  then  $KB \models Q$**

- If  $Q$  is derived from  $KB$  using a given set of rules of inference, then  $Q$  is entailed by  $KB$
- Hence, inference produces only real entailments, or any sentence that follows deductively from the premises is valid

**Completeness: If  $KB \models Q$  then  $KB \vdash Q$**

- If  $Q$  is entailed by  $KB$ , then  $Q$  can be derived from  $KB$  using the rules of inference
- Hence, inference produces all entailments, or all valid sentences can be proved from the premises

# **Problems with Propositional Logic**



# Propositional logic: pro and con

- Advantages
  - Simple KR language sufficient for some problems
  - Lays the foundation for higher logics (e.g., FOL)
  - Reasoning is decidable, though NP complete, and efficient techniques exist for many problems
- Disadvantages
  - Not expressive enough for most problems
  - Even when it is, it can very “un-concise”

# PL is a weak KR language

- Hard to identify “individuals” (e.g., Mary, 3)
- Can’t directly talk about properties of individuals or relations between individuals (e.g., “Bill is tall”)
- Generalizations, patterns, regularities can’t easily be represented (e.g., “all triangles have 3 sides”)
- First-Order Logic (FOL) is expressive enough to represent this kind of information using relations, variables and quantifiers, e.g.,
  - *Every elephant is gray:*  $\forall x (\text{elephant}(x) \rightarrow \text{gray}(x))$
  - *There is a white alligator:*  $\exists x (\text{alligator}(X) \wedge \text{white}(X))$

# PL Example

- Consider the problem of representing the following information:
  - Every person is mortal.
  - Confucius is a person.
  - Confucius is mortal.
- How can these sentences be represented so that we can infer the third sentence from the first two?

# PL Example

- In PL we have to create propositional symbols to stand for all or part of each sentence, e.g.:

P = “person”; Q = “mortal”; R = “Confucius”

- The above 3 sentences are represented as:

P    Q; R    P; R    Q

- The 3rd sentence is entailed by the first two, but we need an explicit symbol, R, to represent an individual, Confucius, who is a member of the classes *person* and *mortal*
- Representing other individuals requires introducing separate symbols for each, with some way to represent the fact that all individuals who are “people” are also “mortal”

# Hunt the Wumpus domain

- Some atomic propositions:  
 S12 = There is a stench in cell (1,2)  
 B34 = There is a breeze in cell (3,4)  
 W22 = Wumpus is in cell (2,2)  
 V11 = We've visited cell (1,1)  
 OK11 = Cell (1,1) is safe.

...

- Some rules:

- (R1) S11      W11      W12      W21  
 (R2) S21      W11      W21      W22      W31  
 (R3) S12      W11      W12      W22      W13  
 (R4) S12      W13      W12      W22      W11

...

- The lack of variables requires us to give similar rules for each cell!

1,4	2,4	3,4	4,4
1,3 W!	2,3	3,3	4,3
1,2 A S OK	2,2 OK	3,2	4,2
1,1 V OK	2,1 B V OK	3,1 P!	4,1

- A = Agent
- B = Breeze
- G = Glitter, Gold
- OK = Safe square
- P = Pit
- S = Stench
- V = Visited
- W = Wumpus

# After the third move

We can prove that the Wumpus is in (1,3) using the four rules given.

See R&N section 7.5

1,4	2,4	3,4	4,4
1,3 W!	2,3	3,3	4,3
1,2 A S OK	2,2 OK	3,2	4,2
1,1 V OK	2,1 B V OK	3,1 P!	4,1

**A** = Agent  
B = Breeze  
G = Glitter, Gold  
OK = Safe square  
P = Pit  
S = Stench  
V = Visited  
W = Wumpus

# Proving W13

Apply MP with S11 and R1:

W11      W12      W21

Apply And-Elimination to this, yielding 3 sentences:

W11,    W12,    W21

Apply MP to  $\sim$ S21 and R2, then apply And-elimination:

W22,    W21,    W31

Apply MP to S12 and R4 to obtain:

W13    W12    W22    W11

Apply Unit resolution on (W13    W12    W22    W11) and W11:

W13    W12    W22

Apply Unit Resolution with (W13    W12    W22) and W22:

W13    W12

Apply UR with (W13    W12) and W12:

W13

QED

# Propositional Wumpus hunter problems

- Lack of variables prevents stating more general rules
  - We need a set of similar rules for each cell
- Change of the KB over time is difficult to represent
  - Standard technique is to index facts with the time when they're true
  - This means we have a separate KB for every time point



# Propositional logic summary

- Inference is the process of deriving new sentences from old
  - **Sound** inference derives true conclusions given true premises
  - **Complete** inference derives all true conclusions from a set of premises
- A **valid sentence** is true in all worlds under all interpretations
- If an implication sentence can be shown to be valid, then—given its premise—its consequent can be derived
- Different logics make different **commitments** about what the world is made of and what kind of beliefs we can have
- **Propositional logic** commits only to the existence of facts that may or may not be the case in the world being represented
  - Simple syntax and semantics suffices to illustrate the process of inference
  - Propositional logic can become impractical, even for very small worlds

# Knowledge Representation

## First Order Logic

## ● Propositional Logic (PL)

A proposition is a statement - which in English is a declarative sentence and Logic defines the ways of putting symbols together to form sentences that represent facts. Every proposition is either true or false. Propositional logic is also called boolean algebra.

**Examples:** (a) The sky is blue., (b) Snow is cold. , (c)  $12 * 12=144$

**Propositional logic :** It is fundamental to all logic.

‡ Propositions are "Sentences"; either true or false but not both.

‡ A sentence is smallest unit in propositional logic

‡ If proposition is true, then truth value is "true"; else "false"

‡ **Example ; Sentence** "Grass is green";

**Truth value** " true";

**Proposition** "yes"

## ■ Statement, Variables and Symbols

**Statement** : A simple statement is one that does not contain any other statement as a part. A compound statement is one that has two or more simple statements as parts called components.

**Operator or connective** : Joins simple statements into compounds, and joins compounds into larger compounds.

### Symbols for connectives

<b>assertion</b>	<b>P</b>					"p is true"
<b>nagation</b>	<b><math>\neg p</math></b>	<b><math>\sim</math></b>	<b>!</b>		<b>NOT</b>	"p is false"
<b>conjunction</b>	<b><math>p \wedge q</math></b>	<b><math>\cdot</math></b>	<b>&amp;&amp;</b>	<b>&amp;</b>	<b>AND</b>	"both p and q are true"
<b>disjunction</b>	<b><math>P \vee q</math></b>	<b>  </b>	<b> </b>		<b>OR</b>	"either p is true, or q is true, or both "
<b>implication</b>	<b><math>p \rightarrow q</math></b>	<b><math>\supset</math></b>	<b><math>\Rightarrow</math></b>		<b>if . . then</b>	"if p is true, then q is true" " p implies q "
<b>equivalence</b>	<b><math>\leftrightarrow</math></b>	<b><math>\equiv</math></b>	<b><math>\Leftrightarrow</math></b>		<b>if and only if</b>	"p and q are either both true or both false"

## ■ Truth Value

The truth value of a statement is its truth or falsity ,

**p** is either true or false,

**$\sim p$**  is either true or false,

**$p \vee q$**  is either true or false, and so on.

"**T**" or "**1**" means "**true**". and

"**F**" or "**0**" means "**false**"

Truth table is a convenient way of showing relationship between several propositions. The truth table for negation, conjunction, disjunction, implication and equivalence are shown below.

<b>p</b>	<b>q</b>	<b><math>\neg p</math></b>	<b><math>\neg q</math></b>	<b><math>p \wedge q</math></b>	<b><math>p \vee q</math></b>	<b><math>p \rightarrow q</math></b>	<b><math>p \leftrightarrow q</math></b>	<b><math>q \rightarrow p</math></b>
<b>T</b>	<b>T</b>	<b>F</b>	<b>F</b>	<b>T</b>	<b>T</b>	<b>T</b>	<b>T</b>	<b>T</b>
<b>T</b>	<b>F</b>	<b>F</b>	<b>T</b>	<b>F</b>	<b>T</b>	<b>F</b>	<b>F</b>	<b>T</b>
<b>F</b>	<b>T</b>	<b>T</b>	<b>F</b>	<b>F</b>	<b>T</b>	<b>T</b>	<b>F</b>	<b>F</b>
<b>F</b>	<b>F</b>	<b>T</b>	<b>T</b>	<b>F</b>	<b>F</b>	<b>T</b>	<b>T</b>	<b>T</b>

# PL is a weak KR language

- Hard to identify “individuals” (e.g., Mary, 3)
- Can’t directly talk about properties of individuals or relations between individuals (e.g., “Bill is tall”)
- Generalizations, patterns, regularities can’t easily be represented (e.g., “all triangles have 3 sides”)
- First-Order Logic (FOL) is expressive enough to represent this kind of information using relations, variables and quantifiers, e.g.,
  - *Every elephant is gray:*  $\forall x (\text{elephant}(x) \rightarrow \text{gray}(x))$
  - *There is a white alligator:*  $\exists x (\text{alligator}(X) \wedge \text{white}(X))$

# Introduction to FOL/FOPL

- Whereas propositional logic assumes the world contains **facts**,
- first-order logic (like natural language) assumes the world contains
  - **Objects**: people, houses, numbers, colors, baseball games, wars, ...
  - **Properties**: red, round, large, even, oval...
  - **Relations**: bigger-than, outside, part-of, occurs-after, owns, visits, comes between, ...
  - **Functions**: father of, best friend, one more than, plus, ...

# Syntax of FOL: Basic Elements

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



# Examples

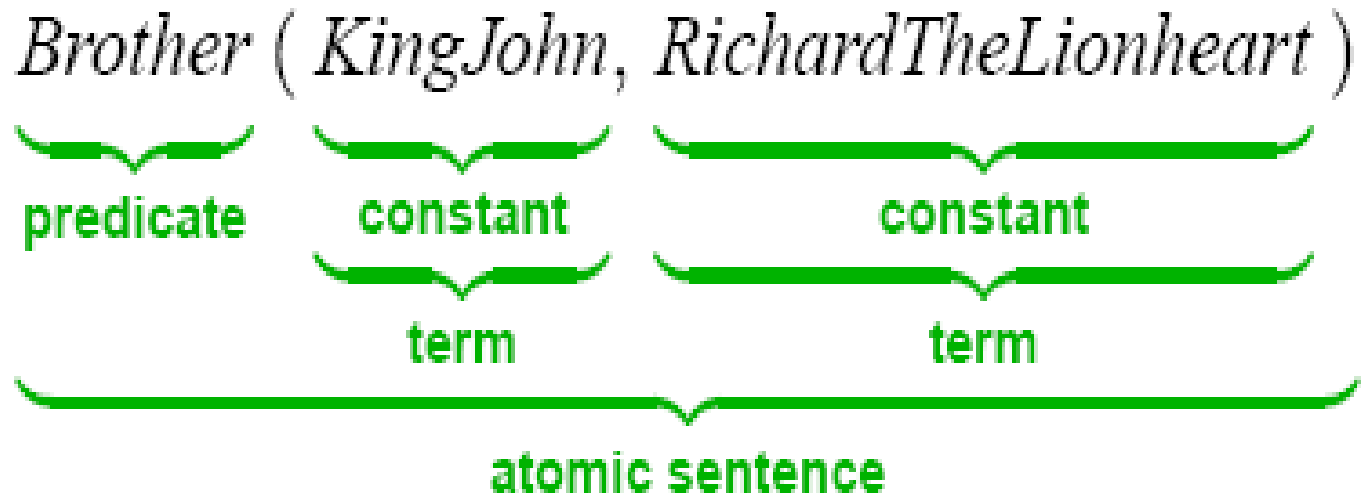
- King John and Richard are brothers

*Brother(KingJohn, Richard)*

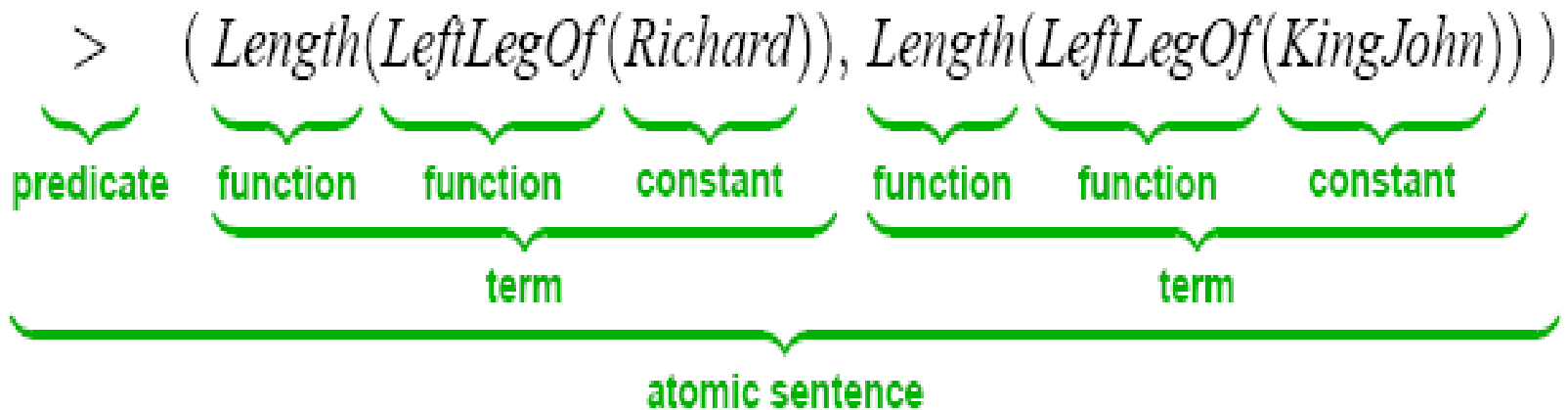
- The length of left leg of Richard is greater than the length of left leg of King John

*>(Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))*

# Atomic Sentences



# Atomic Sentences



# Complex Sentences

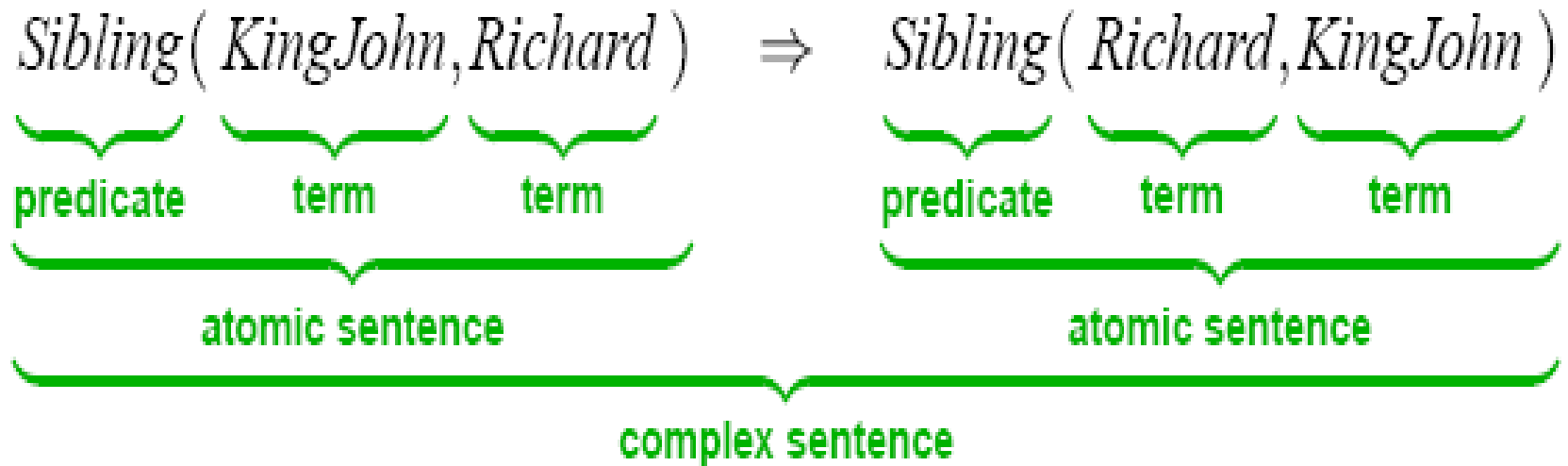
- Complex sentences are made from atomic sentences using connectives:

*S, S1 S2, S1 S2, S1 S2, S1 S2,*

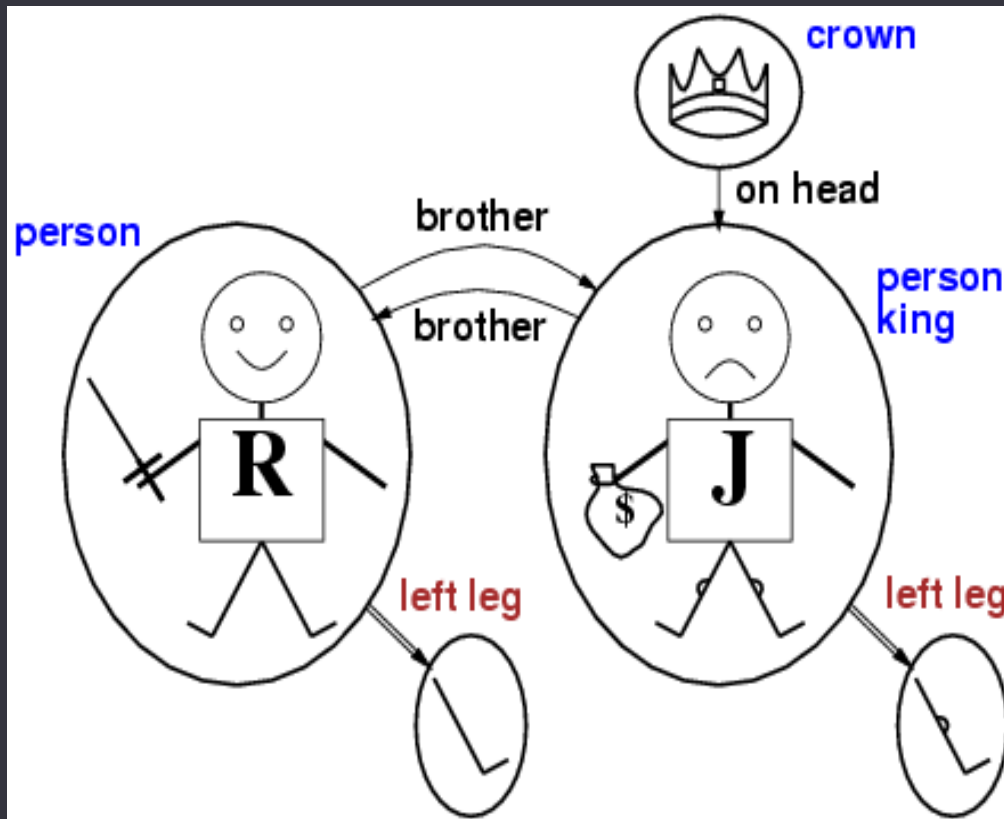
Example

*Sibling(KingJohn, Richard) Sibling(Richard, KingJohn)*

# Complex Sentences

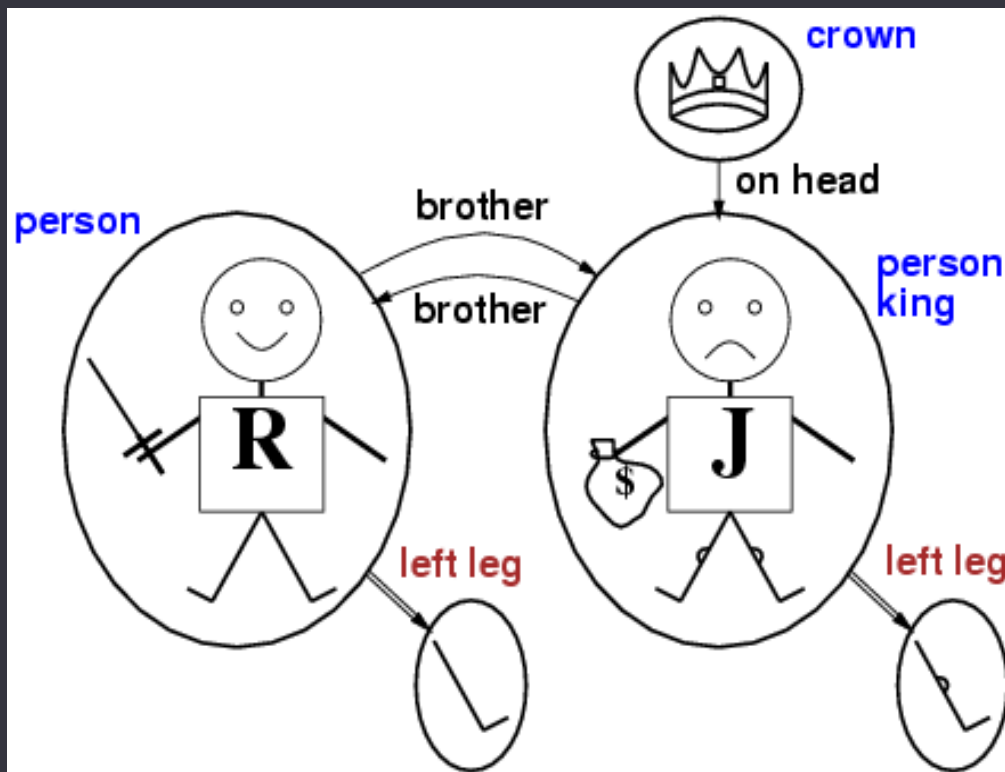


# FOL Illustrated



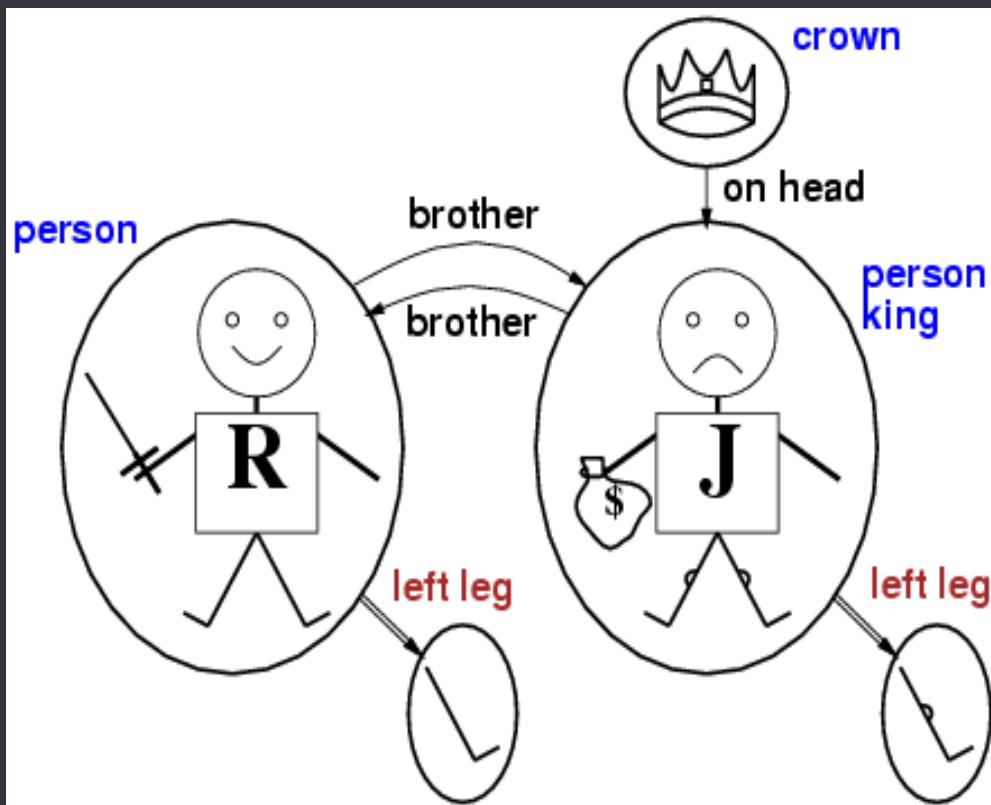
- Five objects-
- 1. Richard the Lionheart
- 2. Evil King John
- 3. Left leg of Richard
- 4. Left leg of John
- 5. The crown

# FOL illustrated



- Objects are related with Relations
- For example, King John and Richard are related with Brother relationship
- This relationship can be denoted by  
     $\text{Brother}(\text{Richard}, \text{John}),$   
     $\text{Brother}(\text{John}, \text{Richard})$

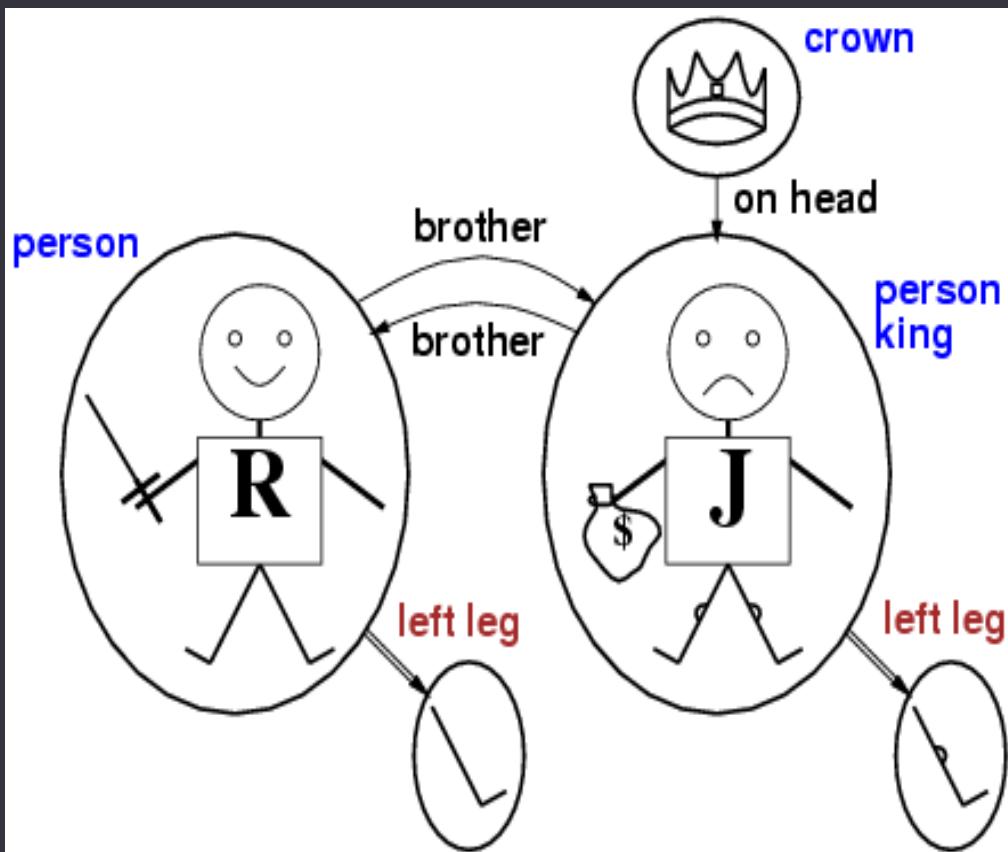
# FOL illustrated



- Again, the crown and King John are related with OnHead Relationship-  
OnHead (Crown,John)
- Brother and OnHead are binary relations as they relate two of objects.

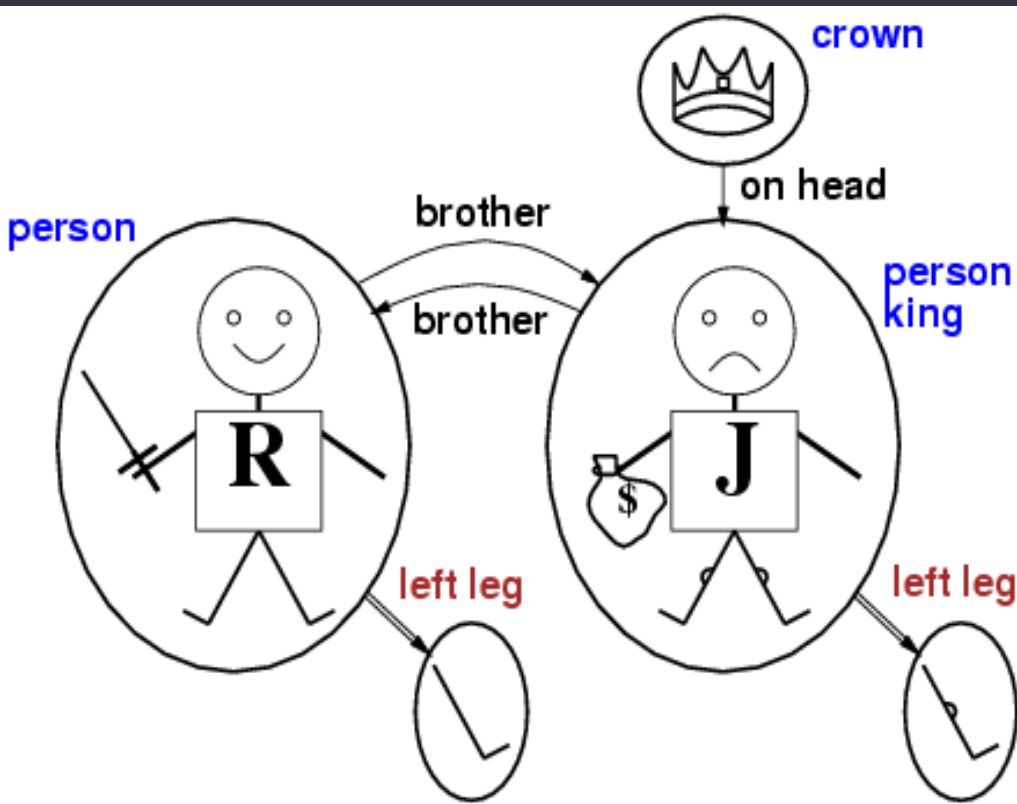


# FOL illustrated



- Properties are relations that are unary.
- In this case, Person can be such property acting upon both Richard and John
  - Person (Richard)
  - Person (John)
- Again, king can be acted only upon John
  - King (John)

# FOL illustrated



- Certain relationships are best performed when expressed as functions.
- Means one object is related with exactly one object.

Richard -> Richard's left leg

John -> John's left leg

# Universal quantification

- $\langle \text{variables} \rangle \langle \text{sentence} \rangle$ 
  - $\forall x P(x)$
- Translated into the English language, the expression is understood as:
  - "For all  $x$ ,  $P(x)$  holds",
  - "for each  $x$ ,  $P(x)$  holds" or
  - "for every  $x$ ,  $P(x)$  holds"
- "All cars have wheels" could be transformed into the propositional form,  $\forall x P(x)$ 
  - $P(x)$  is the predicate denoting:  **$x$  has wheels**, and
  - the universe of discourse is only populated by cars. **Where  $x$  is car**

# Universal quantification

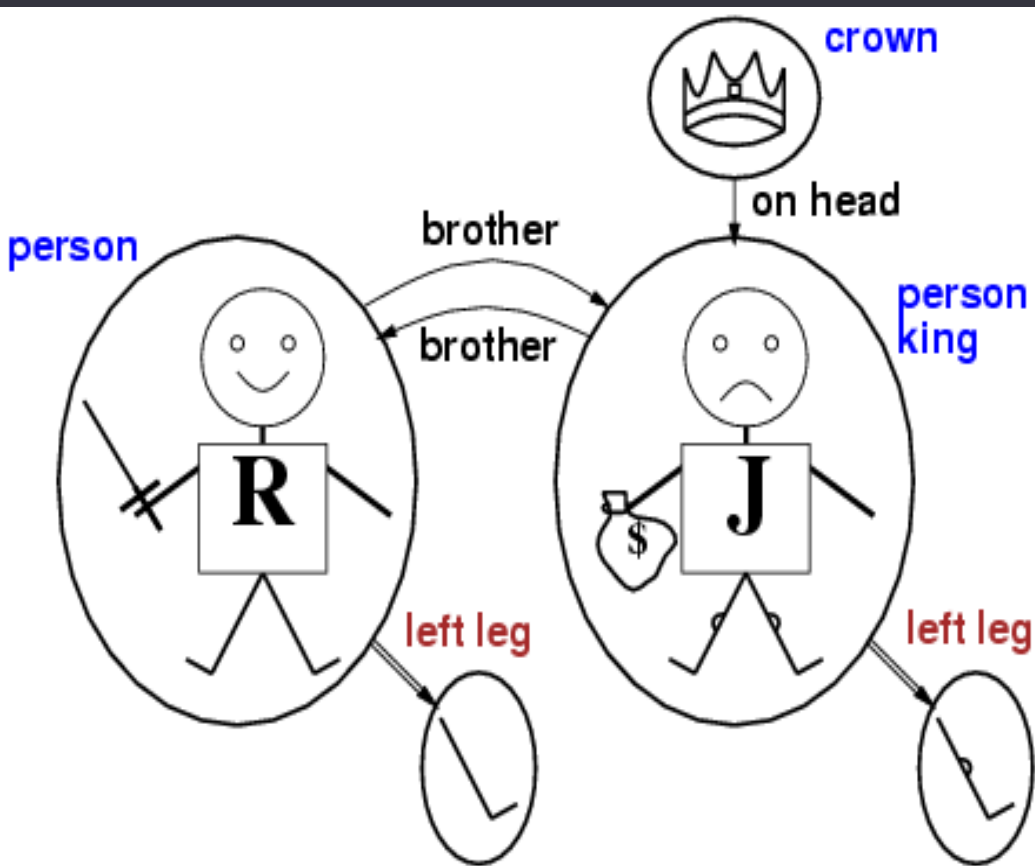
- If all the elements in the universe of discourse can be listed then the universal quantification  $\forall x P(x)$  is equivalent to the conjunction:

$$P(x_1) \wedge P(x_2) \wedge P(x_3) \wedge \dots \wedge P(x_n) .$$

For example, in the above example of  $\forall x P(x)$ , if we knew that there were **only** 4 cars in our universe of discourse ( $c_1$ ,  $c_2$ ,  $c_3$  and  $c_4$ ) then we could also translate the statement as:

$$P(c_1) \wedge P(c_2) \wedge P(c_3) \wedge P(c_4)$$

# Universal quantification

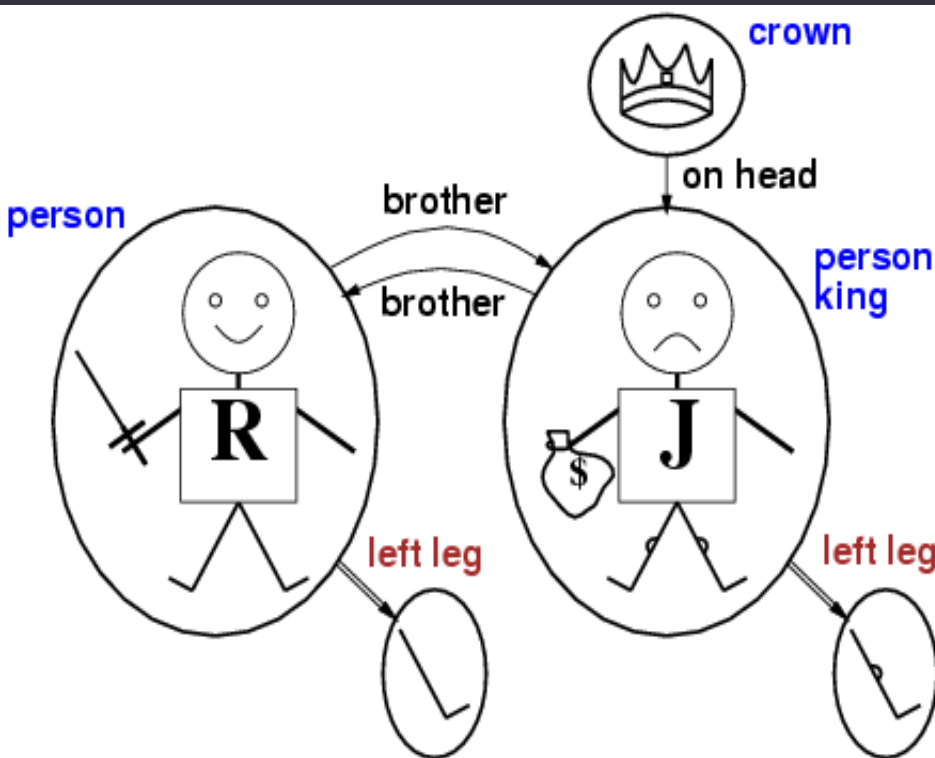


- Remember, we had five objects, let us replace them with a variable  $x$ -

  - $x \rightarrow$  Richard the Lionheart
  - $x \rightarrow$  Evil King John
  - $x \rightarrow$  Left leg of Richard
  - $x \rightarrow$  Left leg of John
  - $x \rightarrow$  The crown

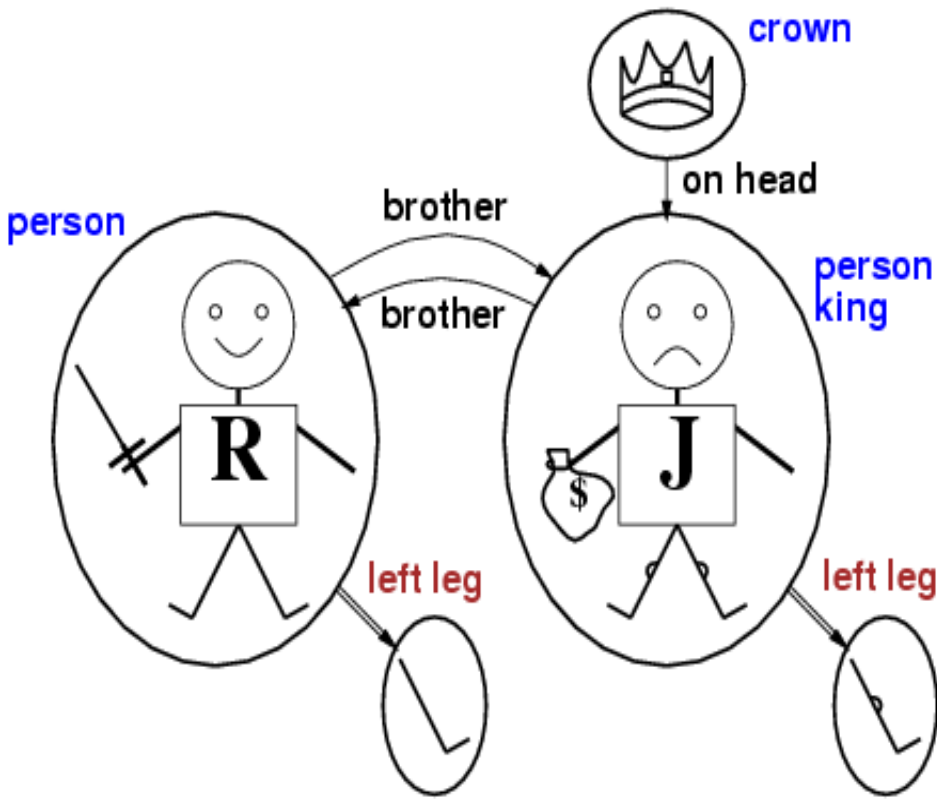
# Universal quantification

- Now, for the quantified sentence  
 $\forall x \text{ King}(x) \supset \text{Person}(x)$



Richard is king      Richard is Person  
John is king      John is person  
Richard's left leg is king      Richard's left leg  
is person  
John's left leg is king      John's left leg is  
person  
The crown is king      the crown is person

# Universal quantification



# Existential quantification

- $\langle \text{variables} \rangle \langle \text{sentence} \rangle$ 
  - $\exists x P(x)$
- Translated into the English language, the expression is understood as:
  - "There exists an  $x$  such that  $P(x)$ "
  - "There is at least one  $x$  such that  $P(x)$ "
- "*Someone likes you*" could be transformed into the propositional form,  $\exists x P(x)$ 
  - $P(x)$  is the predicate meaning:  **$x$  likes you**,
  - The universe of discourse contains (but is not limited to) all living creatures.



# Existential quantification

- If all the elements in the universe of discourse can be listed, then the existential quantification  $\exists x P(x)$  is equivalent to the disjunction:

$$P(x_1) \vee P(x_2) \vee P(x_3) \vee \dots \vee P(x_n)$$

For example, in the above example of  $\exists x P(x)$ , if we knew that there were **only** 5 living creatures in our universe of discourse (say: me, he, she, they and we), then we could also write the statement as:

$$P(\text{me}) \vee P(\text{he}) \vee P(\text{she}) \vee P(\text{they}) \vee P(\text{we})$$

# Order of application of quantifiers

- When more than one variables are quantified in a wff such as  $\exists y \forall x P(x, y)$ , they are applied from the inside, that is, the one closest to the atomic formula is applied first.
- Thus  $\exists y \forall x P(x, y)$  reads  $\exists y [\forall x P(x, y)]$ , and we say "there exists a  $y$  such that for every  $x$ ,  $P(x, y)$  holds" or "for some  $y$ ,  $P(x, y)$  holds for every  $x$ ".
- WFF = Well formed Formula

# Order of application of quantifiers

- The positions of the same type of quantifiers can be switched without affecting the truth value as long as there are no quantifiers of the other type between the ones to be interchanged.
- For example  $\exists x \exists y \exists z P(x, y, z)$  is equivalent to  $\exists y \exists x \exists z P(x, y, z)$ ,  $\exists z \exists y \exists x P(x, y, z)$ , etc.
- It is the same for the universal quantifier.

# Order of application of quantifiers

- However, the positions of different types of quantifiers can **not** be switched.
- For example  $\exists x \exists y P(x, y)$  is **not** equivalent to  $\exists y \exists x P(x, y)$ .

# Order of application of quantifiers

- $\forall x \exists y x < y$

- “for every number  $x$ , there is a number  $y$  that is greater than  $x$ ”

- $\exists y \forall x x < y$

- “there is a number that is greater than every (any) number”

# Properties of quantifiers

- $\exists x \exists y$  is the same as  $\exists y \exists x$
- $\forall x \forall y$  is the same as  $\forall y \forall x$
- $\exists x \forall y$  is **not** the same as  $\forall y \exists x$

# Properties of quantifiers

**Quantifier duality:** each can be expressed using the other

- $\exists x \text{ Likes}(x, \text{IceCream})$  is equivalent to  
•  $\neg \forall x \neg \text{ Likes}(x, \text{IceCream})$
- $\forall x \text{ Likes}(x, \text{Broccoli})$  is equivalent to  
•  $\neg \exists x \neg \text{ Likes}(x, \text{Broccoli})$

# Properties of quantifiers

- Equivalences-

1.  $\exists x P$  is equivalent to  $\exists x \neg \neg P$

2.  $\exists x \neg P$  is equivalent to  $\neg \forall x P$

3.  $\forall x \neg P$  is equivalent to  $\neg \exists x P$

4.  $\forall x P$  is equivalent to  $\neg \exists x \neg P$



Let  $E(x)$  mean  $x$  is even and  $G(x, y)$  mean  $x > y$ . Let the universe be the set of naturals.

Let  $\forall$  represent the universal and  $\exists$  the existential quantifiers, respectively.

$\forall x \exists y G(y, x)$  is true, but  $\exists x \forall y G(y, x)$  is false.

Yes    No

$\exists x E(x)$  is true.

Yes    No

$\forall x \forall y G(x, y)$  is true.

Yes    No

$\forall x G(\exists y, x)$  is a proposition.

Yes    No

# Example knowledge base

- The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.
- Prove that Col. West is a criminal

# Example knowledge base

... it is a crime for an American to sell weapons to hostile nations:  
*American(x) Weapon(y) Sells(x,y,z) Hostile(z) Criminal(x)*

Nono ... has some missiles,

*Owns(Nono,x)*

*Missile(x)*

... all of its missiles were sold to it by Colonel West  
*Missile(x) Owns(Nono,x) Sells(West,x,Nono)*

Missiles are weapons:

*Missile(x) Weapon(x)*

An enemy of America counts as "hostile":

*Enemy(x,America) Hostile(x)*

West, who is American ...

*American(West)*

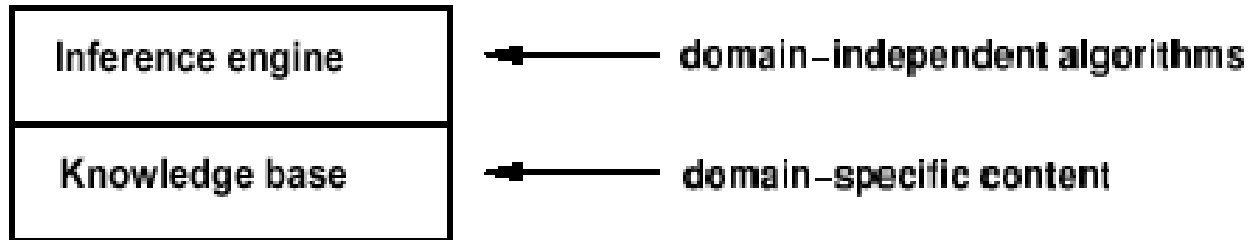
The country Nono, an enemy of America ...

*Enemy(Nono,America)*

# **Artificial Intelligence**

## **Knowledge and Reasoning**

# Knowledge Base



Knowledge base = set of sentences in a formal language

Declarative approach to building an agent (or other system):

TELL it what it needs to know

Then it can ASK itself what to do—answers should follow from the KB

Agents can be viewed at the knowledge level

i.e., what they know, regardless of how implemented

Or at the implementation level

i.e., data structures in KB and algorithms that manipulate them

# Representation

- Good knowledge representation should combine  
= natural language + formal language
- In this chapter we concentrate on first-order logic (FOL), which forms the basis of most representations schemes in AI.

# General Definition

- ***Logic*** → It is a formal language representing information that conclusions can be easily drawn
- ***Syntax*** → defines the sentences in the language
- ***Semantic*** → defines the meaning of the sentences

# Type of Logic

Logics are characterized by what they commit to as “primitives”

Ontological commitment: what exists—facts? objects? time? beliefs?

Epistemological commitment: what states of knowledge?

Language	Ontological commitment (what exists in the world)	Epistemological commitment (what an agent believes)
<b>Propositional logic</b>	<b>Facts</b>	<b>True/false/unknown</b>
<b>First-Order Logic</b>	<b>Facts,object,relations</b>	<b>True/false/unknown</b>
<b>Temporal logic</b>	<b>Facts,object,relations, times</b>	<b>True/false/unknown</b>
<b>Probability</b>	<b>Facts</b>	<b>Degree of belief 0..1</b>



# 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

If  $S_1$  and  $S_2$  is a sentence,  $S_1 \wedge S_2$  is a sentence

If  $S_1$  and  $S_2$  is a sentence,  $S_1 \vee S_2$  is a sentence

If  $S_1$  and  $S_2$  is a sentence,  $S_1 \Rightarrow S_2$  is a sentence

If  $S_1$  and  $S_2$  is a sentence,  $S_1 \Leftrightarrow S_2$  is a sentence

# Propositional Logic

## *Semantic*

Each model specifies true/false for each proposition symbol

E.g.  $A \quad B \quad C$   
*True True False*

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 <u>and</u>	$S_2$	is true		
$S_1 \vee S_2$	is true iff	$S_1$	is true <u>or</u>	$S_2$	is true		
$S_1 \Rightarrow S_2$	is true iff	$S_1$	is false <u>or</u>	$S_2$	is true		
	i.e.,		is false iff	$S_1$	is true <u>and</u>	$S_2$	is false
$S_1 \Leftrightarrow S_2$	is true iff	$S_1 \Rightarrow S_2$	is true <u>and</u>	$S_2 \Rightarrow S_1$	is true		

# Inference

- Process by which conclusions are reached
- Logical inference: is a process that implements the entailment(**deduction/conclusion**) relation between sentences

$KB \vdash_i \alpha$  = sentence  $\alpha$  can be derived from  $KB$  by procedure  $i$

Soundness:  $i$  is sound **Or truth-preserving**

whenever  $KB \vdash_i \alpha$ , it is also true that  $KB \models \alpha$

Completeness:  $i$  is complete if

whenever  $KB \models \alpha$ , it is also true that  $KB \vdash_i \alpha$

## *Propositional Inference: Enumeration Method*

Let  $\alpha = A \vee B$  and  $KB = (A \vee C) \wedge (B \vee \neg C)$

Is it the case that  $KB \models \alpha$ ?

Check all possible models— $\alpha$  must be true wherever  $KB$  is true

<i>A</i>	<i>B</i>	<i>C</i>	$A \vee C$	$B \vee \neg C$	$KB$	$\alpha$
<i>False</i>	<i>False</i>	<i>False</i>				
<i>False</i>	<i>False</i>	<i>True</i>				
<i>False</i>	<i>True</i>	<i>False</i>				
<i>False</i>	<i>True</i>	<i>True</i>				
<i>True</i>	<i>False</i>	<i>False</i>				
<i>True</i>	<i>False</i>	<i>True</i>				
<i>True</i>	<i>True</i>	<i>False</i>				
<i>True</i>	<i>True</i>	<i>True</i>				

*We check only if  $KB$  is true*



# Normal Forms \*

Other approaches to inference use syntactic operations on sentences, often expressed in standardized forms

Conjunctive Normal Form (CNF—universal)

*conjunction of disjunctions of literals*  
*clauses*

E.g.,  $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

Disjunctive Normal Form (DNF—universal)

*disjunction of conjunctions of literals*  
*terms*

E.g.,  $(A \wedge B) \vee (A \wedge \neg C) \vee (A \wedge \neg D) \vee (\neg B \wedge \neg C) \vee (\neg B \wedge \neg D)$

# Validity & Satisfiability

A sentence is valid if it is true in all models

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

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 true in no models

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

Satisfiability is connected to inference via the following:

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

i.e., prove  $\alpha$  by *reductio ad absurdum*

# Standard Logical Equivalences

1.  $A \wedge A \equiv A$
2.  $A \vee A \equiv A$
3.  $A \wedge (B \wedge C) \equiv (A \wedge B) \wedge C$
4.  $A \vee (B \vee C) \equiv (A \vee B) \vee C$  [  $\vee$  is associative ]
5.  $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$  [  $\wedge$  is distributive over  $\vee$  ]
6.  $A \vee (A \wedge B) \equiv A$
7.  $A \wedge (A \vee B) \equiv A$
9.  $A \wedge \text{true} \equiv A$
10.  $A \vee \text{false} \equiv A$
11.  $A \wedge \text{true} \equiv A$
12.  $A \vee \text{false} \equiv A$
13.  $A \Rightarrow B \equiv \neg A \vee B$  [implication elimination]
14.  $\neg(A \wedge B) \equiv \neg A \vee \neg B$  [De Morgan]
15.  $\neg(A \vee B) \equiv \neg A \wedge \neg B$  [De Morgan]



# Seven Inference Rules for Propositional Logic

1. Modus-Ponens or Implication elimination (*From an implication and the premise of the implication, you can infer the conclusion*)

$$\Rightarrow, \quad \vdash$$

2. And-Elimination (*From a conjunction, you can infer any of the conjuncts*)

$$\begin{array}{cccc} 1 & 2 & 3\dots & \vdash i \\ n & & & \end{array}$$

3. And-Introduction (*From a list of sentences, you can infer their conjunction*)

$$1, 2, 3\dots \vdash 1 \quad 2 \quad 3.. \quad n$$

4. Or-Introduction (*From a sentence, you can infer its disjunction*)

$$i \vdash 1 \quad 2 \quad 3.. \quad n$$

# Seven Inference Rules for Propositional Logic

## 5. Double-Negation Elimination

$$\frac{}{\neg\neg A \vdash A}$$

## 6. Unit Resolution (*From disjunction, if one is false, then you can infer the other one is true*)

$$\frac{A \vee B, \neg A}{B}$$

## 7. Resolution (*Because cannot be true and false in the same time*)

$$\frac{A \vee B, \neg A \vee C}{B \vee C}$$

# Example (1)

- $\{A, A\} \vdash$  (prove ?)
  - 1.  $A \Leftrightarrow A$  (using truth table)
  - 2.  $A$  (I will replace  $A$ )
  - 3.  $A, A$  (I will add from KB  $A$ )
- 
- ( Elimination)

$\vdash$

# Example (2)

- $\{A \quad B\} \vdash \{A \quad B\}$  (prove ?)
  1.  $A \quad B$  (assumption)
  2.  $A, B$  (by elimination)

---

$A \quad B$  (by introduction)

# Example (3)

$\vdash (A \rightarrow B) \rightarrow (B \rightarrow A)$

1.  $A$  (assumption)
2.  $A \rightarrow B$  (assumption)
3.  $B$  (by modus ponens)
4.  $B, B \rightarrow A$  (introduce  $B$  by assumption)
- 5.
6.  $A \rightarrow B$  (reduction by absurdum)
7.  $(A \rightarrow B) \rightarrow (B \rightarrow A)$  (introduction)

$(A \rightarrow B) \rightarrow (B \rightarrow A)$

---

# Example (4) \*

$\vdash$  (A B) ((B C) ((C D) (A D))) ?

1. -----
2. -----
3. -----
4. -----
5. -----
6. -----
7. -----
8. -----
9. -----

# Inference using rules

- To proof  $KB \models A$
- Write  $KB \quad A$  in CNF Form
- Apply inference rules to find contradiction

# First Order Logic



# Definition

- General-purpose representation language that is based on an ontological commitment to the existence of the objects and relations in the world.
- World → consists of:
  - **Objects:** *people, houses, numbers, colors, wares*
  - **Relations:** *brother of , bigger than, inside, part of*
  - **Properties:** *red, round, long, short,,,*
  - **Functions:** *father of, best friend, one more than ,,,*

# Example

- “One Plus Two Equals Three ”
  - *Objects: One, Two, Three, One Plus Two*
  - *Relations: Equals*
  - *Functions: Plus*
- “Congratulation Letter written with Blue Pen”
  - *Objects: Letter, Pen*
  - *Relations: written with*
  - *Properties: Blue, Congratulation*

# Syntax & Semantic

- ***In FOL = Sentences + Terms (which represents objects)***
- ***Sentences*** → are built using quantifiers and predicate symbols
- ***Terms*** → are built using constants, variables and functions symbol.

Sentence		<b>AtomicSentence</b>   Sentence <b>Connective</b> Sentence   Quantifier Var,,,,,Sentence   Sentence   (Sentence)
AtomicSentence		<b>Predicate(Term,,,,)</b>   Term = Term
Term		<b>Function( Term,,)</b>   Constant   Variable
Connective		=>       ⇔
Quantifier		.   .
Constant		<b>A</b>   1   3   John   Riad,,,,
Variable		<b>a</b>   b   c   x   y   z
Predicate		<b>Before</b>   HasColor  After
Function		<b>Mother</b>   LeftLegOf   Equal

# Syntax and Semantic

## *Predicate Symbol*

It is a particular relation in the model between pair of objects  $\rightarrow$  *Predicate(Term,,,,)*

$\langle (1,2) \quad \rangle (3,4) \quad \text{Brother}(\text{mohamed}, \text{Mostefa})$

## *Function Symbol*

A given object it is related to exactly one other object by the relation  $\rightarrow$  *Function(Term,,,,)*

*FatherOf(Ahmad) \quad \quad \quad \text{Equal}(Plus(1,2))*

# Syntax and Semantic

## *Terms*

It is an expression that refers to an object →

*Function(Term,,,)* | *variable* | *constant*

*FatherOf( Khalid) x y 2 Riyadh Ahmad*

## *Atomic Sentence*

Is formed from a predicate symbol followed by a parenthesized list of terms.

*Predicate(Term,,,)* or *term =term*

*Older(Youssef, 30)*

*1 = 1*

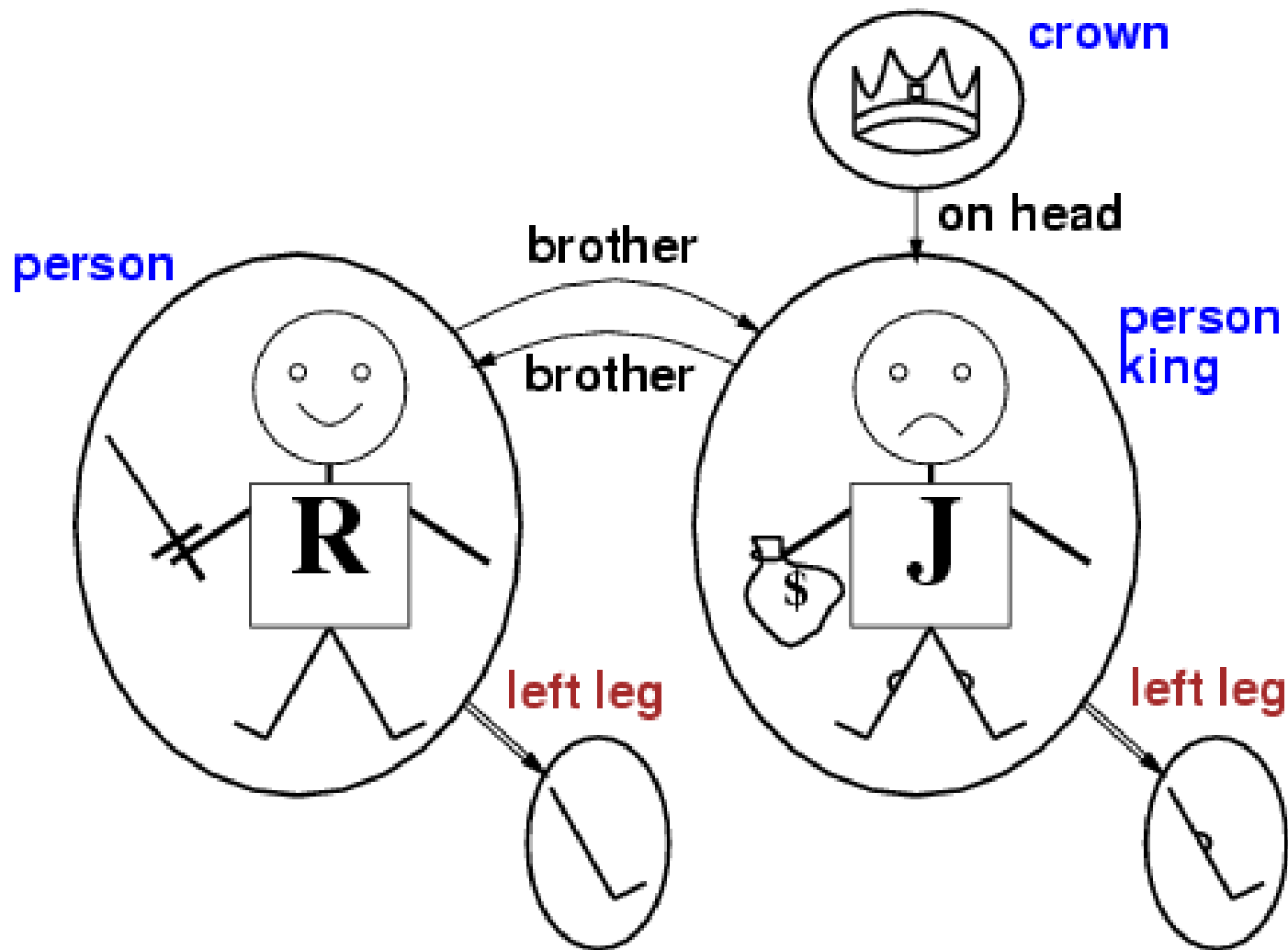
# Syntax and Semantic

## *Complex sentences*

We can use logical connective to construct more complex sentences

$S1$	$S1 \ S2$	$S1 \ S2$	$S1 \Rightarrow S2$	$S1 \Leftrightarrow S2$
$> (1,2)$	$(1,2)$			
$> (1,2)$	$>(1,2)$			

# Model in FOL





# Syntax and Semantic

## *Universal Quantifier*

(variables), (Sentence)

Everyone at PSU is smart →

$$\forall x \text{ At}(x, \text{PSU}) \Rightarrow \text{Smart}(x)$$

∇ **P is conjunction of instantiations of P**

$$\text{At}(\text{mohamed}, \text{PSU}) \Rightarrow \text{Smart}(\text{mohamed})$$

$$\text{At}(\text{Khalid}, \text{PSU}) \Rightarrow \text{Smart}(\text{Khalid})$$

▶ **!** *The implies ( $\Rightarrow$ ) is the main connective with*

*$\forall x \text{ At}(x, \text{PSU}) \quad \text{Smart}(x)$  will has different meaning:  
“everyone is at PSU and everyone is smart”*

# Syntax and Semantic

## *Existential Quantifier*

(variables), (Sentence)

Someone in PSU is smart →

$\exists x \text{ At}(x, \text{PSU}) \wedge \text{Smart}(x)$

**P is disjunctions of instantiations of P**

$\text{At}(\text{mohamed}, \text{PSU}) \wedge \text{Smart}(\text{mohamed})$

$\text{At}(\text{Khalid}, \text{PSU}) \wedge \text{Smart}(\text{Khalid})$

► *! The and (  $\wedge$  ) is the main connective with  $\exists x \text{ At}(x, \text{PSU}) \Rightarrow \text{Smart}(x)$  will have different meaning: "The sentence is True for anyone who is not in PSU" by using the Rule:  $(A \Rightarrow B) \quad (A \vee B)$*

# Properties of Quantifiers

$\forall x \forall y$  is the same as  $\forall y \forall x$  (why??)

$\exists x \exists y$  is the same as  $\exists y \exists x$  (why??)

$\exists x \forall y$  is not the same as  $\forall y \exists x$

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

“There is a person who loves everyone in the world”

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

“Everyone in the world is loved by at least one person”

Quantifier duality: each can be expressed using the other

$\forall x \text{ Likes}(x, \text{IceCream}) \quad \neg \exists x \neg \text{Likes}(x, \text{IceCream})$

$\exists x \text{ Likes}(x, \text{Broccoli}) \quad \neg \forall x \neg \text{Likes}(x, \text{Broccoli})$

# Sentences in FOL

Brothers are siblings

.

“Sibling” is reflexive

.

One's mother is one's female parent

.

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

# Sentences in FOL

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

.

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

.

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

.

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

# Equality in FOL

$term_1 = term_2$  is true under a given interpretation  
if and only if  $term_1$  and  $term_2$  refer to the same object

E.g.,  $1 = 2$  and  $\forall x \times(Sqrt(x), Sqrt(x)) = x$  are satisfiable  
 $2 = 2$  is valid

E.g., definition of (full) *Sibling* in terms of *Parent*:

$$\forall x, y \text{ Sibling}(x, y) \Leftrightarrow [\neg(x = y) \wedge \exists m, f \neg(m = f) \wedge \\ \text{Parent}(m, x) \wedge \text{Parent}(f, x) \wedge \text{Parent}(m, y) \wedge \text{Parent}(f, y)]$$

# Exercises Using FOL

## ■ Exercise#1:

- *Represent the sentence*

*"There are two only smarts students in KSU"*

$x, y, z$  student( $x$ ), student ( $y$ ), student( $z$ ) and smart( $X$ )  
and sm,art( $y$ ) and smart( $z$ ) and different( $x,y$ ) and  
(equal( $x,z$ ) or equal ( $y,z$ ))

## ■ Exercise#2 (8.11)Page 269

- *Write axioms describing the predicates:*

*"GrandChild – Brother – Sister – Daughter – Son"*

# Problem

- Tariq, Saeed and Yussef belong to the Computer Club.
  - *Every member of the club is either programmer or a analyst or both*
  - *No analyst likes design, and all programmer like C++*
  - *Yussef dislikes whatever Tariq likes and likes whatever Tariq dislikes*
  - *Tariq likes C++ and design*



# Solution

- $S(x)$  means  $x$  is a programmer
- $M(x)$  means  $x$  is a analyst
- $L(x,y)$  means  $x$  likes  $y$

***Is there any member of the club who is analyst but not programmer?***

$$\cdot \quad \exists x \quad S(x) \vee M(x)$$

$$\sim \cdot \quad \exists x \quad M(x) \wedge \neg L(x, \text{design})$$

$$\cdot \quad \forall x \quad S(x) \Rightarrow L(x, \text{C++})$$

$$\cdot \quad \forall y \quad L(\text{Yussef}, y) \Leftrightarrow \neg L(\text{Tariq}, y)$$

$$L(\text{tariq}, \text{C++})$$

$$L(\text{Tariq}, \text{design})$$

# Asking and Getting answers

- To add sentence to a knowledge base KB, we would call

*TELL( KB, **m,c** Mother(c ) =m  $\Leftrightarrow$  Female(m)  
Parent(m,c))*

- To ask the KB:

*ASK( KB, Grandparent(Ahmad,Khalid))*

# Chaining

- Simple methods used by most inference engines to produce a line of reasoning
- Forward chaining: the engine begins with the initial content of the workspace and proceeds toward a final conclusion
- Backward chaining: the engine starts with a goal and finds knowledge to support that goal

# Forward Chaining

## ■ Data driven reasoning

- *bottom up*
- *Search from facts to valid conclusions*

## ■ Given database of true facts

- *Apply all rules that match facts in database*
- *Add conclusions to database*
- *Repeat until a goal is reached, OR repeat until no new facts added*

# Forward Chaining Example

Suppose we have three rules:

R1: If A and B then D

R2: If B then C

R3: If C and D then E

If facts A and B are present, we infer D from R1 and infer C from R2. With D and C inferred, we now infer E from R3.

# Example

## Rules

R1: IF hot AND smoky THEN fire  
R2: IF alarm-beeps THEN smoky  
R3: IF fire THEN switch-sprinkler

## Facts

- alarm-beeps
- hot

First cycle: R2 holds

Second cycle: R1 holds

Third cycle: R3 holds

- smoky

- fire

- switch-sprinkler

*Action*

# Forward Chaining Algorithm

- **Read the initials facts**
- **Begin**
  - Filter Phase => Find the fired rules
  - **While** Fired rules not empty **AND** not end **DO**
    - Choice Phase => Solve the conflicts
    - Apply the chosen rule
    - Modify (if any) the set of rule
  - **End do**
- **End**

# Backward Chaining

- Goal driven reasoning
  - *top down*
  - *Search from hypothesis and finds supporting facts*
- To prove goal G:
  - *If G is in the initial facts, it is proven.*
  - *Otherwise, find a rule which can be used to conclude G, and try to prove each of that rule's conditions.*



# Backward Chaining Example

The same three rules:

R1: If A and B then D

R2: If B then C

R3: If C and D then E

If E is known, then R3 implies C and D are true.

R2 thus implies B is true (from C) and R1 implies A and B are true (from D).

# Example

## Facts

- alarm-beeps
- hot

## Rules

R1: IF hot AND smoky THEN fire  
R2: IF alarm-beeps THEN smoky  
R3: IF fire THEN switch-sprinkler

## Hypothesis

Should I switch the sprinklers on?

## Evidence

IF fire

IF hot ✓  
IF smoky

IF alarm-beeps ✓

Use R3

Use R1

Use R2

# Backward Chaining Algorithm

- Filter Phase
- **IF** set of selected rules is empty **THEN** Ask the user
- **ELSE**
  - **WHILE** not end **AND** we have a selected rules **DO**
    - Choice Phase
    - Add the conditions of the rules
    - **IF** the condition not solved **THEN** put the condition as a goal to solve
  - **END WHILE**

# Application

- Wide use in expert systems
  - *Backward chaining: Diagnosis systems*
    - start with set of hypotheses and try to prove each one, asking additional questions of user when fact is unknown.
  - *Forward chaining: design/configuration systems*
    - see what can be done with available components.

# Comparison

## ■ Backward chaining

- *From hypotheses to relevant facts*
- *Good when:*
  - Limited number of (clear) hypotheses
  - Determining truth of facts is expensive
  - Large number of possible facts, mostly irrelevant

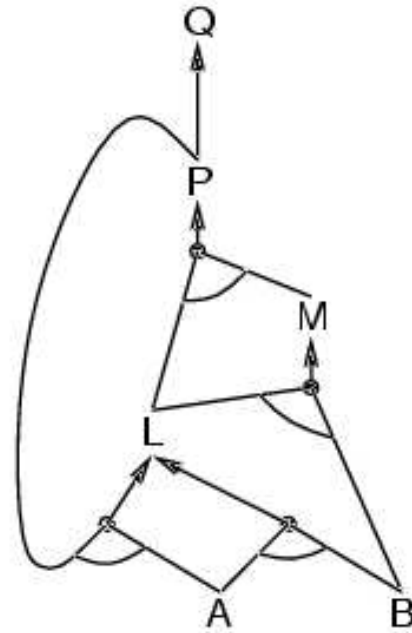
## ■ Forward chaining

- *From facts to valid conclusions*
- *Good when*
  - Less clear hypothesis
  - Very large number of possible conclusions
  - True facts known at start

# Forward chaining

- Idea: fire any rule whose premises are satisfied in the *KB*,
  - *add its conclusion to the KB, until query is found*

$P \Rightarrow Q$   
 $L \wedge M \Rightarrow P$   
 $B \wedge L \Rightarrow M$   
 $A \wedge P \Rightarrow L$   
 $A \wedge B \Rightarrow L$   
 $A$   
 $B$



# Forward chaining algorithm

```
function PL-FC-ENTAILS?(KB, q) returns true or false
  local variables: count, a table, indexed by clause, initially the number of premises
                  inferred, a table, indexed by symbol, each entry initially false
                  agenda, a list of symbols, initially the symbols known to be true

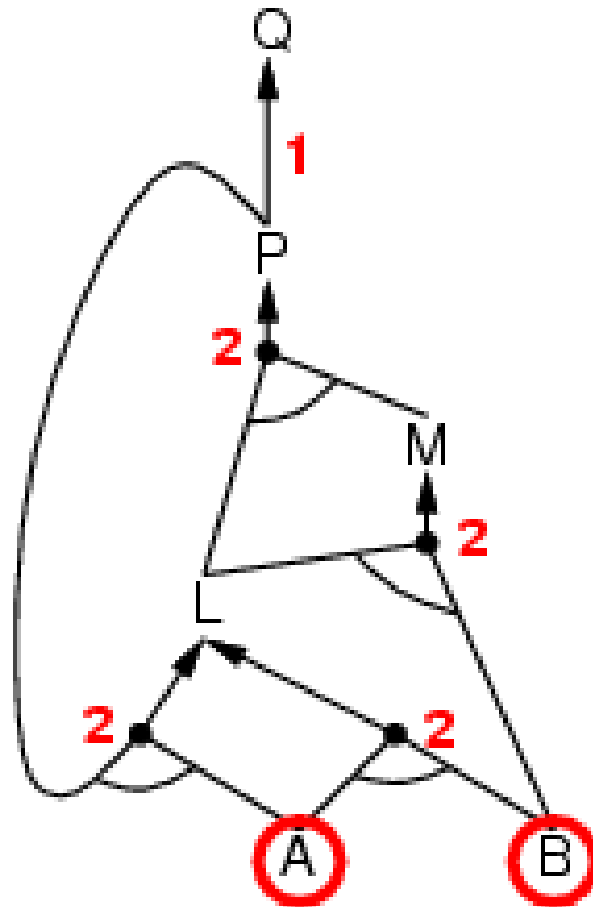
  while agenda is not empty do
    p ← POP(agenda)
    unless inferred[p] do
      inferred[p] ← true
      for each Horn clause c in whose premise p appears do
        decrement count[c]
        if count[c] = 0 then do
          if HEAD[c] = q then return true
          PUSH(HEAD[c], agenda)

  return false
```

Forward chaining is sound and complete for Horn KB

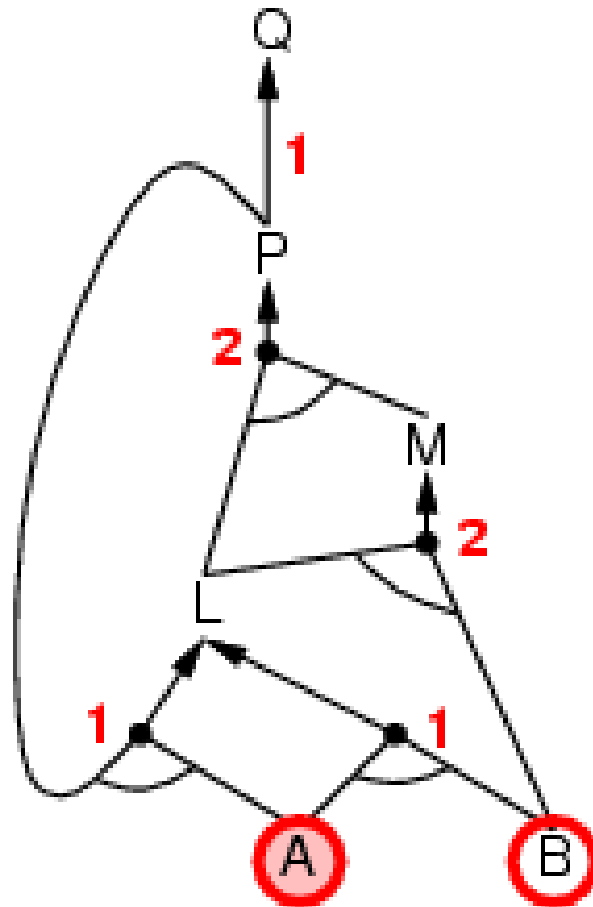


# Forward chaining example

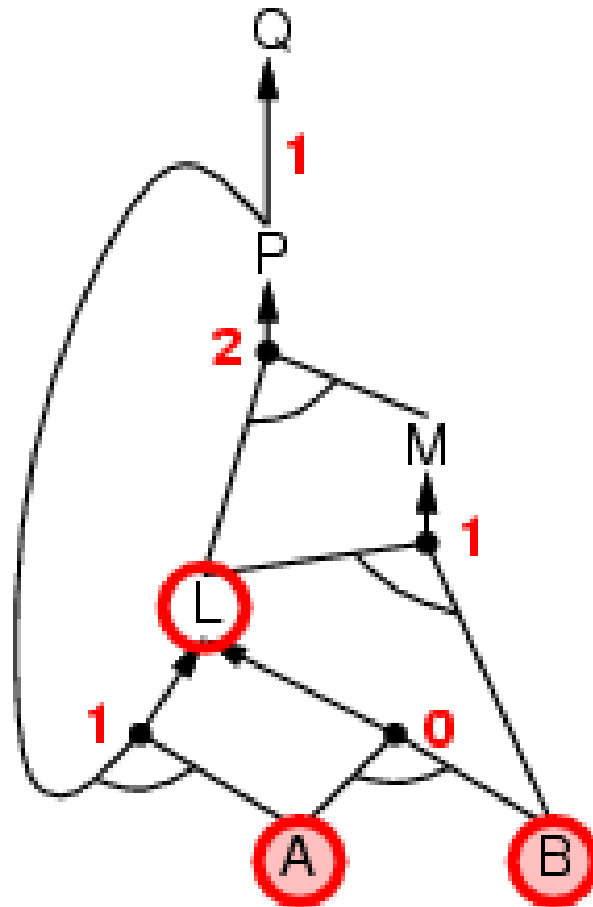




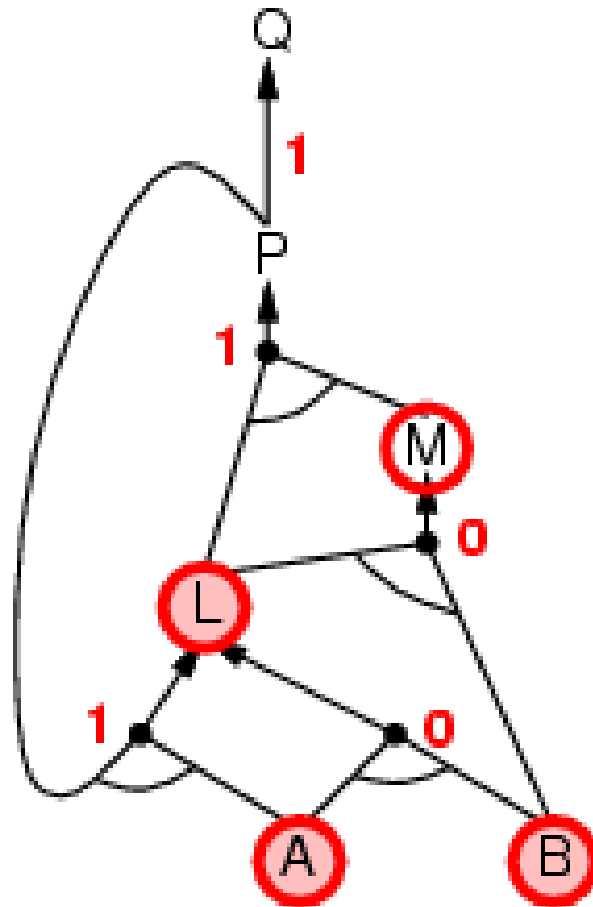
# Forward chaining example



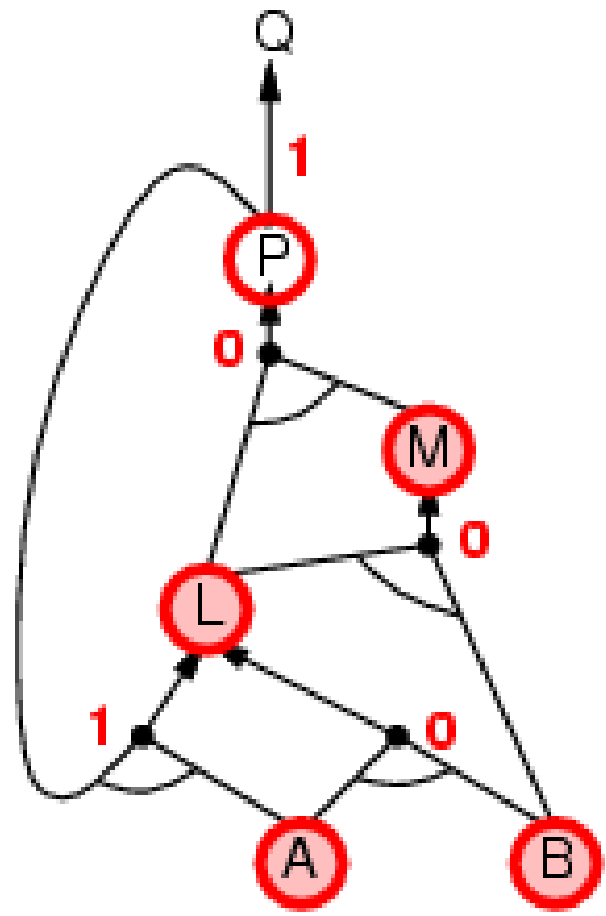
# Forward chaining example



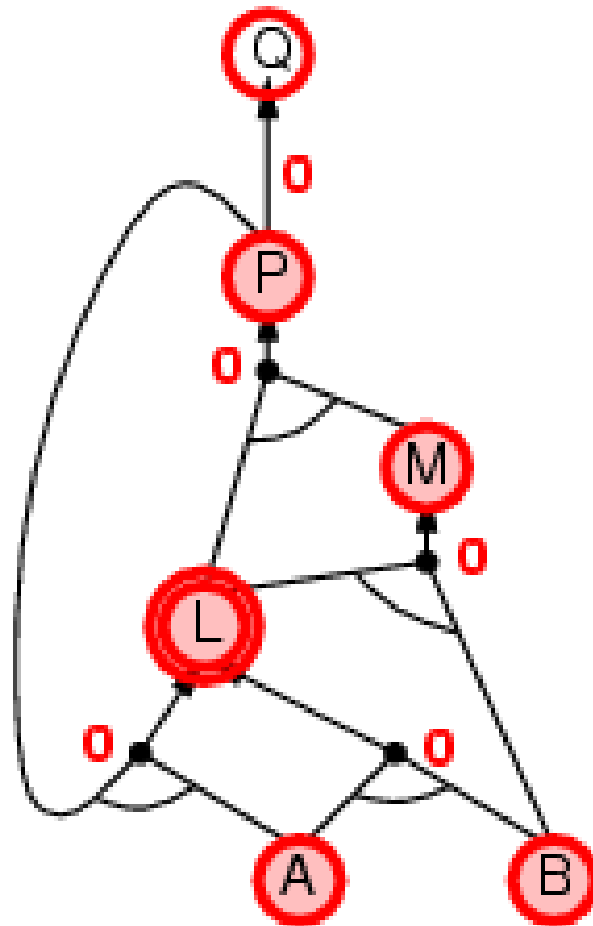
# Forward chaining example



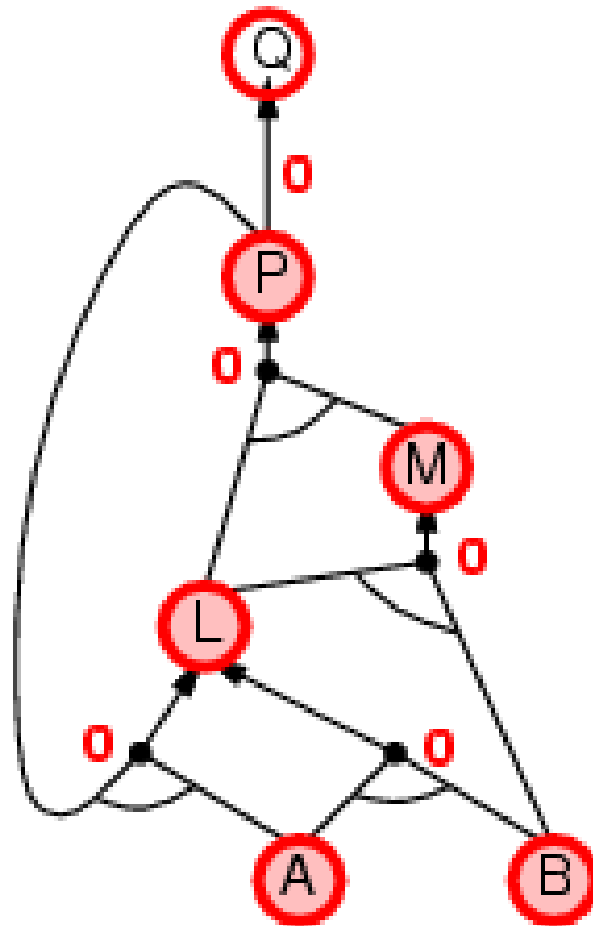
# Forward chaining example



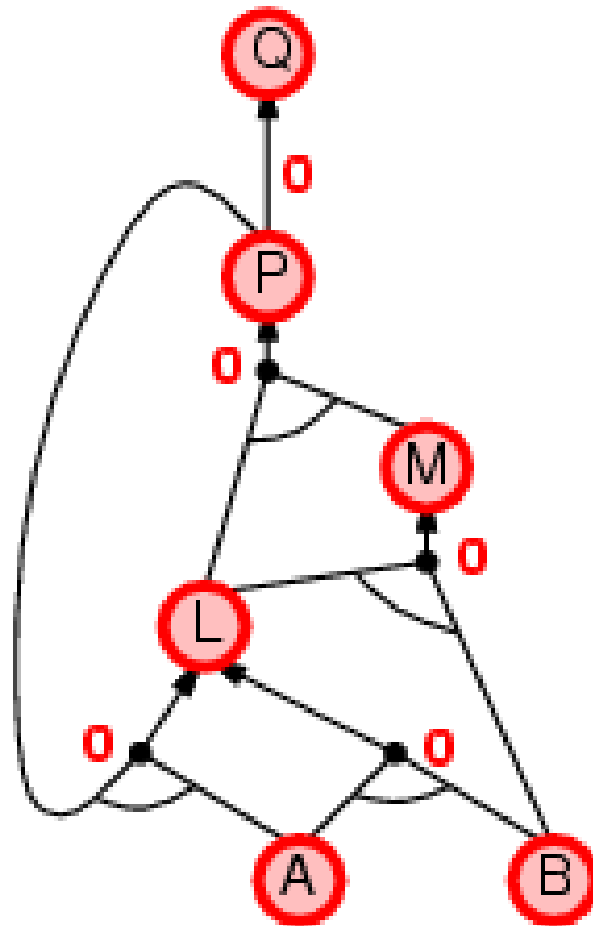
# Forward chaining example



# Forward chaining example



# Forward chaining example



# Backward chaining

Idea: work backwards from the query  $q$ :

*to prove  $q$  by BC,*

check if  $q$  is known already, or

prove by BC all premises of some rule concluding  $q$

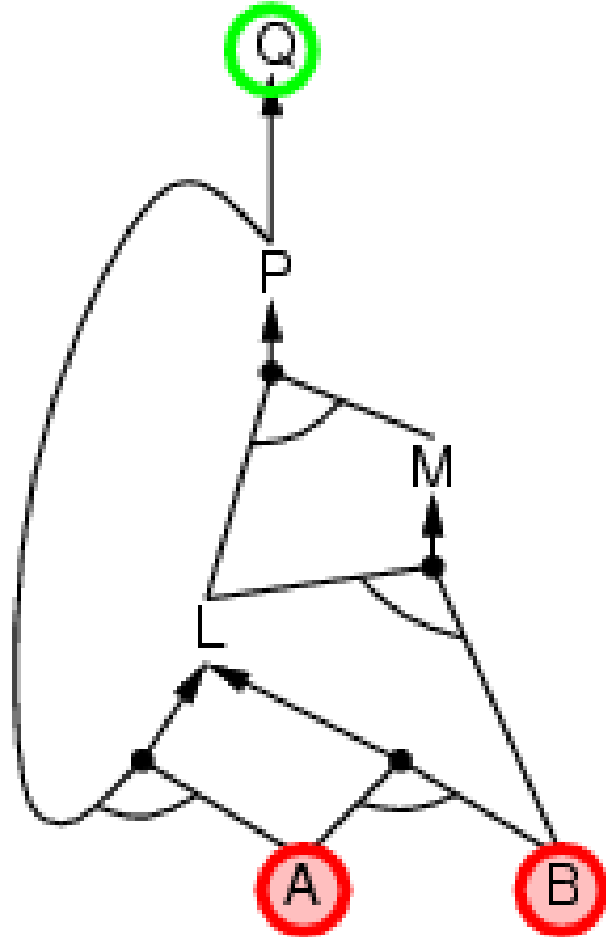
Avoid loops: check if new subgoal is already on the goal stack

Avoid repeated work: check if new subgoal

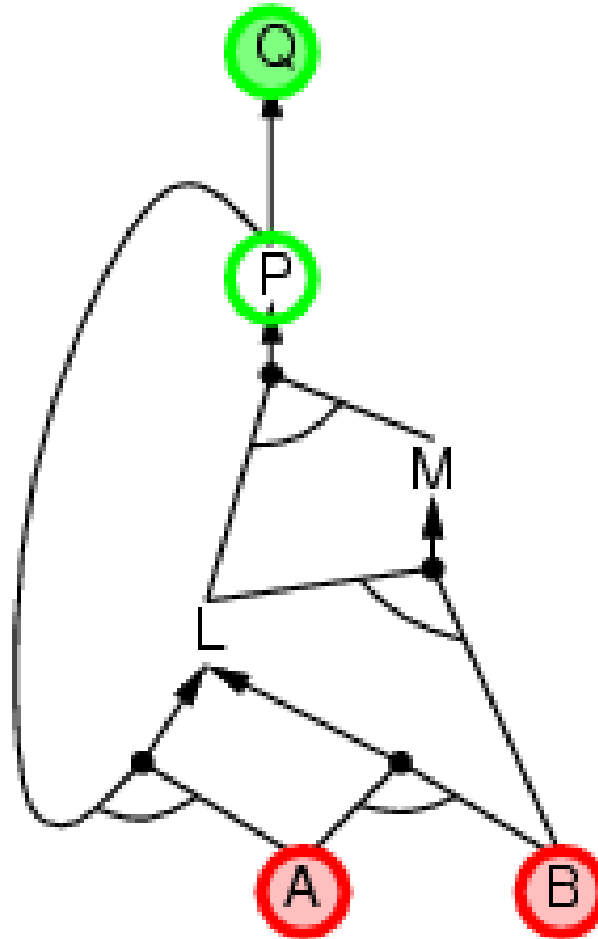
*has already been proved true, or*



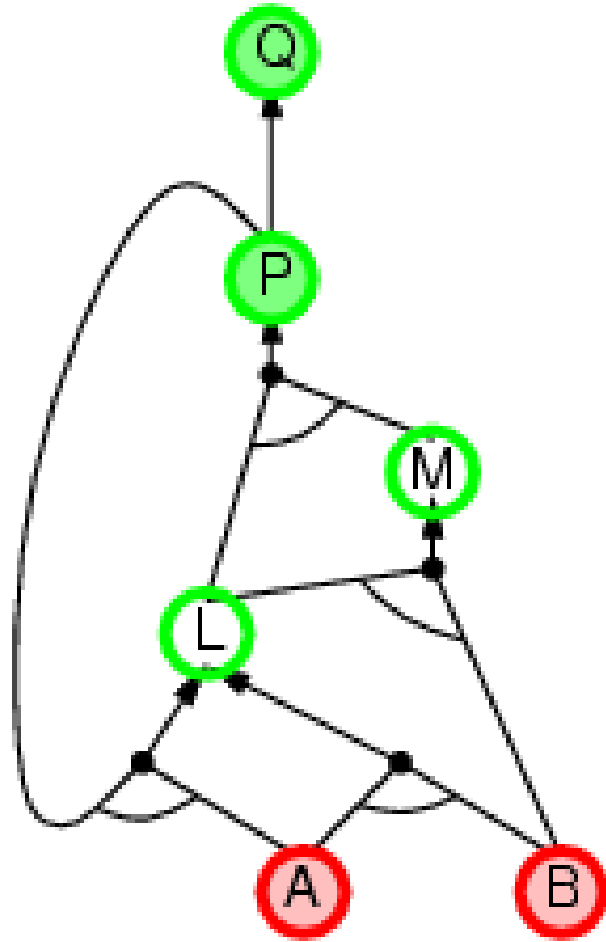
# Backward chaining example



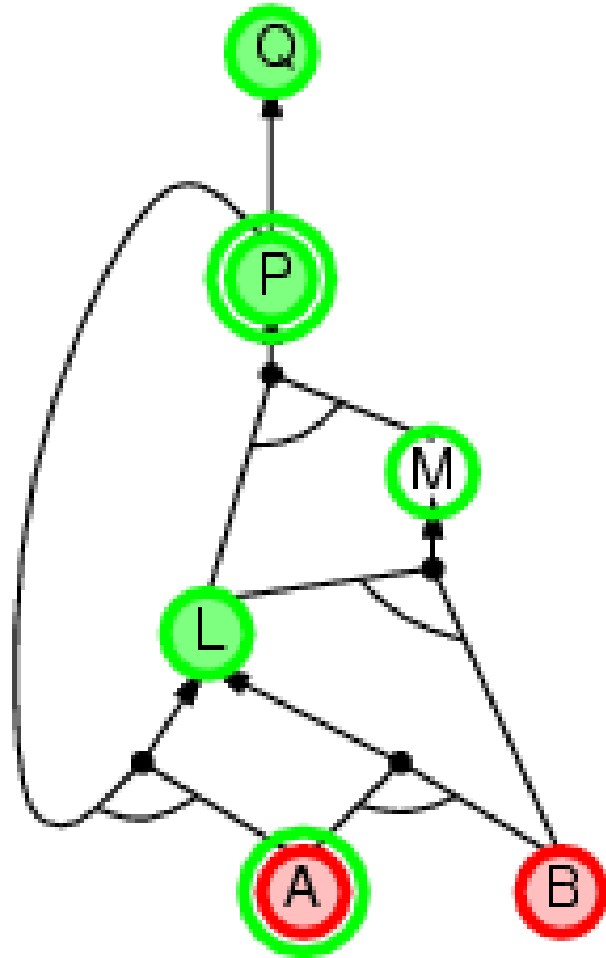
# Backward chaining example



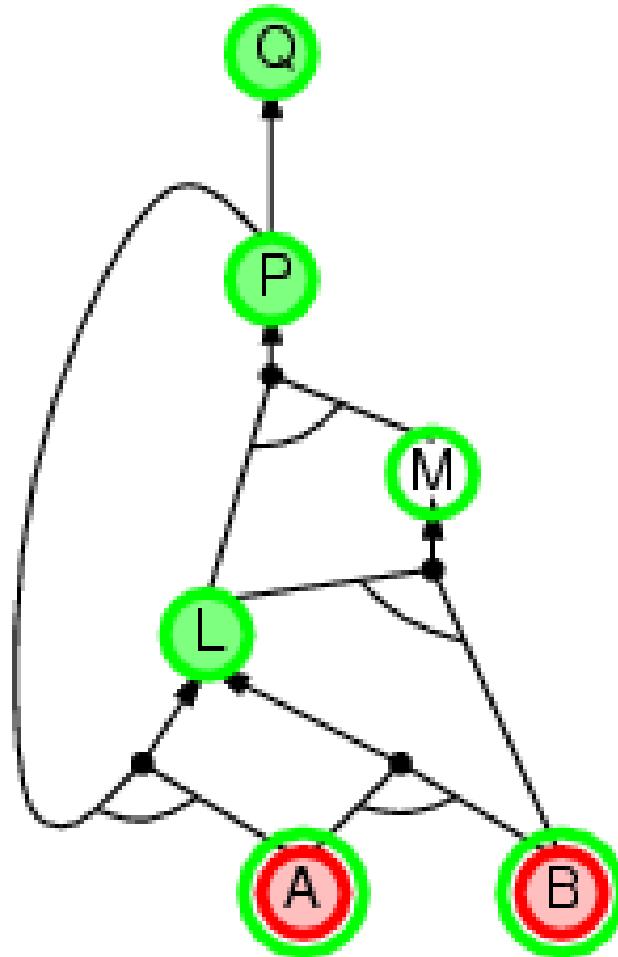
# Backward chaining example



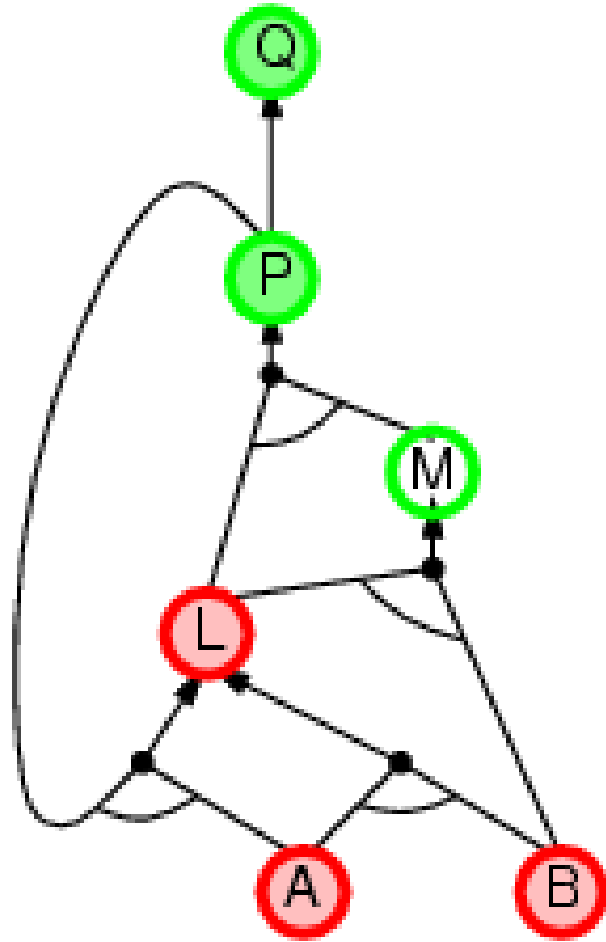
# Backward chaining example



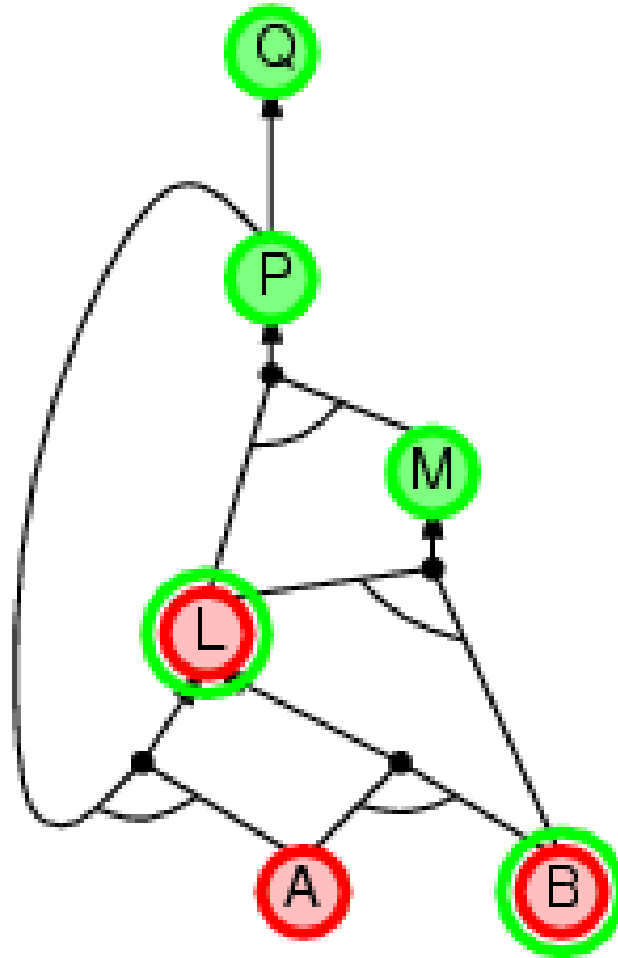
# Backward chaining example



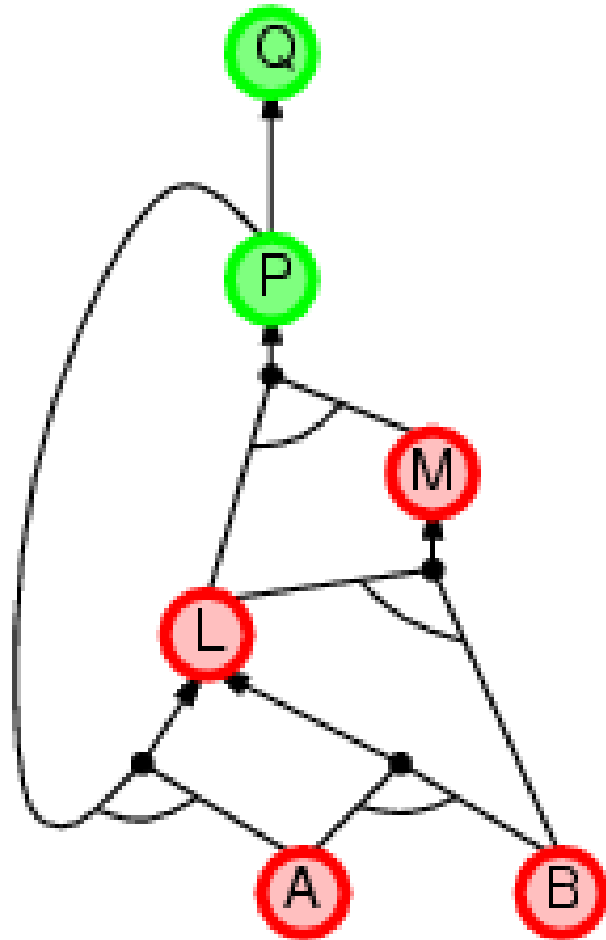
# Backward chaining example



# Backward chaining example

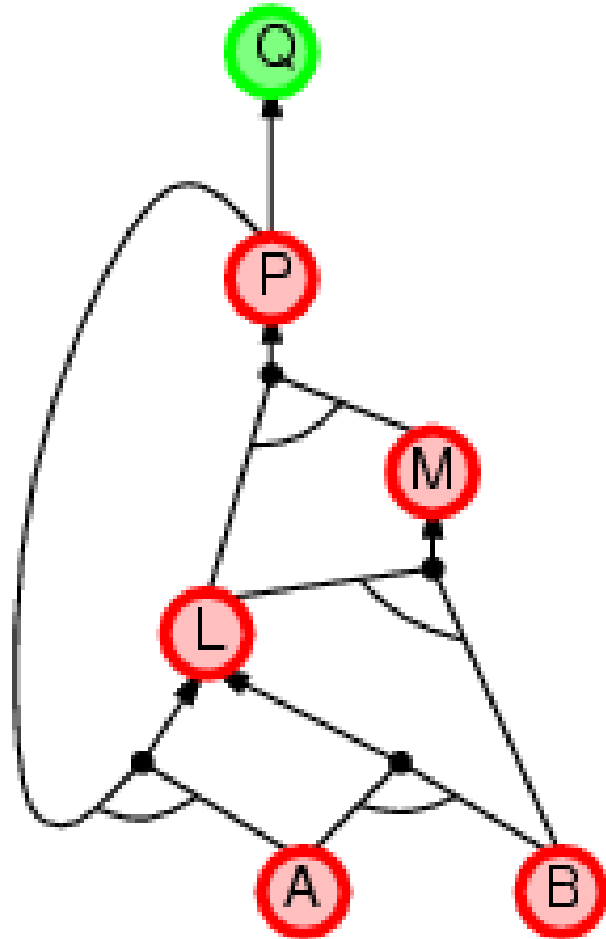


# Backward chaining example

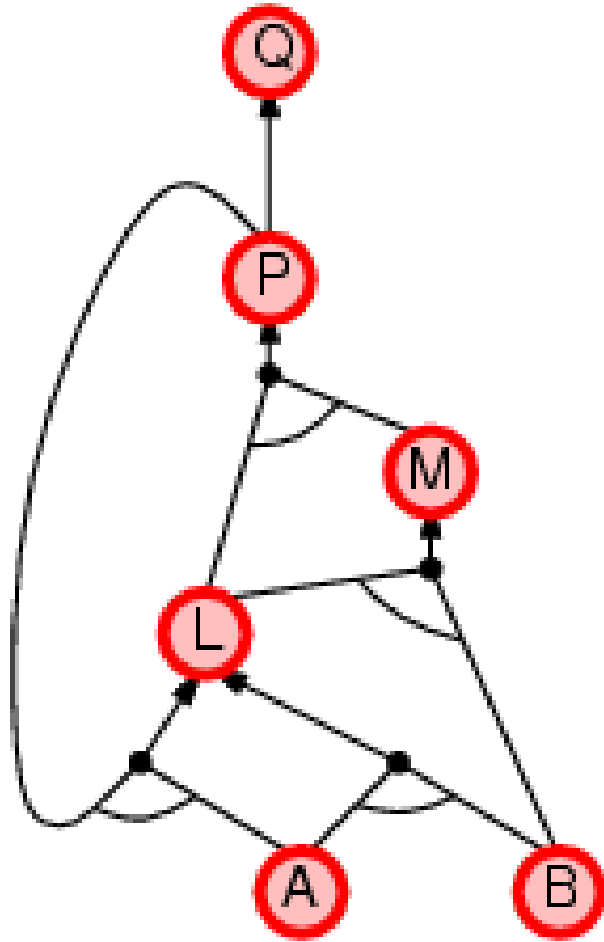




# Backward chaining example



# Backward chaining example



# Forward vs. backward chaining

- FC is data-driven, automatic, unconscious processing,
  - *e.g., object recognition, routine decisions*
    -
- May do lots of work that is irrelevant to the goal
- BC is goal-driven, appropriate for problem-solving,
  - *e.g., Where are my keys? How do I get into a PhD program?*
  -

Complexity of BC can be **much less** than linear in size of KB

■

- Forward and Backward Chaining
- Resolution

# Forward Chaining

- Forward Chaining
  - Start with atomic sentences in the KB and apply Modus Ponens in the forward direction, adding new atomic sentences, until no further inferences can be made.

# Forward Chaining

- Given a new fact, generate all consequences
- Assumes all rules are of the form
  - C1 and C2 and C3 and.... --> Result
- Each rule & binding generates a new fact
- This new fact will “trigger” other rules
- Keep going until the desired fact is generated
- (Semi-decidable as is FOL in general)

# FC: Example Knowledge Base

- The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy America, has some missiles, and all of its missiles were sold to it by Col. West, who is an American.
- Prove that Col. West is a criminal.

# FC: Example Knowledge Base

- ...it is a crime for an American to sell weapons to hostile nations

*American(x) Weapon(y) Sells(x,y,z) Hostile(z) Criminal(x)*

- Nono...has some missiles

*x Owns(Nono, x) Missiles(x)*

*Owns(Nono, M1) and Missile(M1)*

- ...all of its missiles were sold to it by Col. West

*x Missile(x) Owns(Nono, x) Sells(West, x, Nono)*

- Missiles are weapons

*Missile(x) Weapon(x)*



# FC: Example Knowledge Base

- An enemy of America counts as “hostile”  
*Enemy( x, America )      Hostile(x)*
- Col. West who is an American  
*American( Col. West )*
- The country Nono, an enemy of America  
*Enemy(Nono, America)*

# FC: Example Knowledge Base

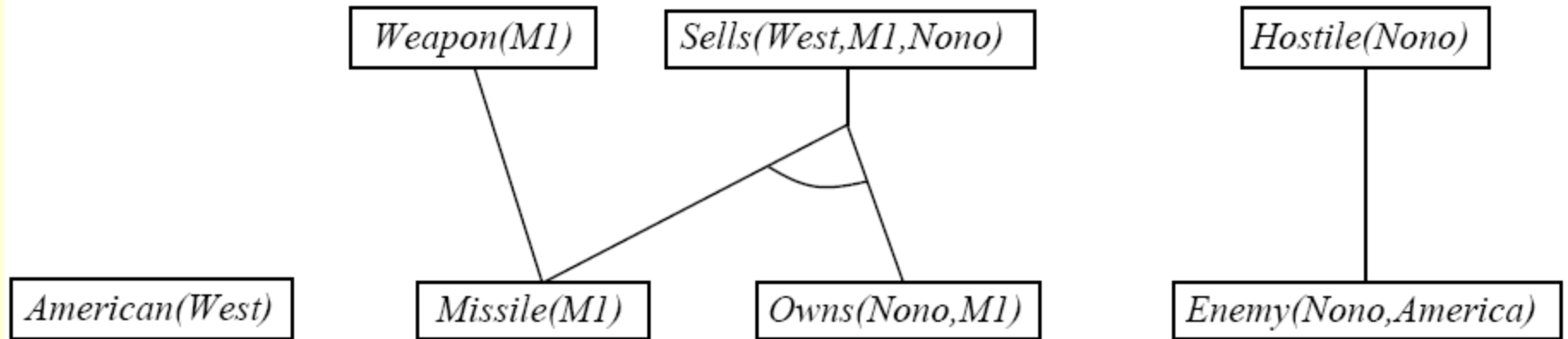
*American(West)*

*Missile(M1)*

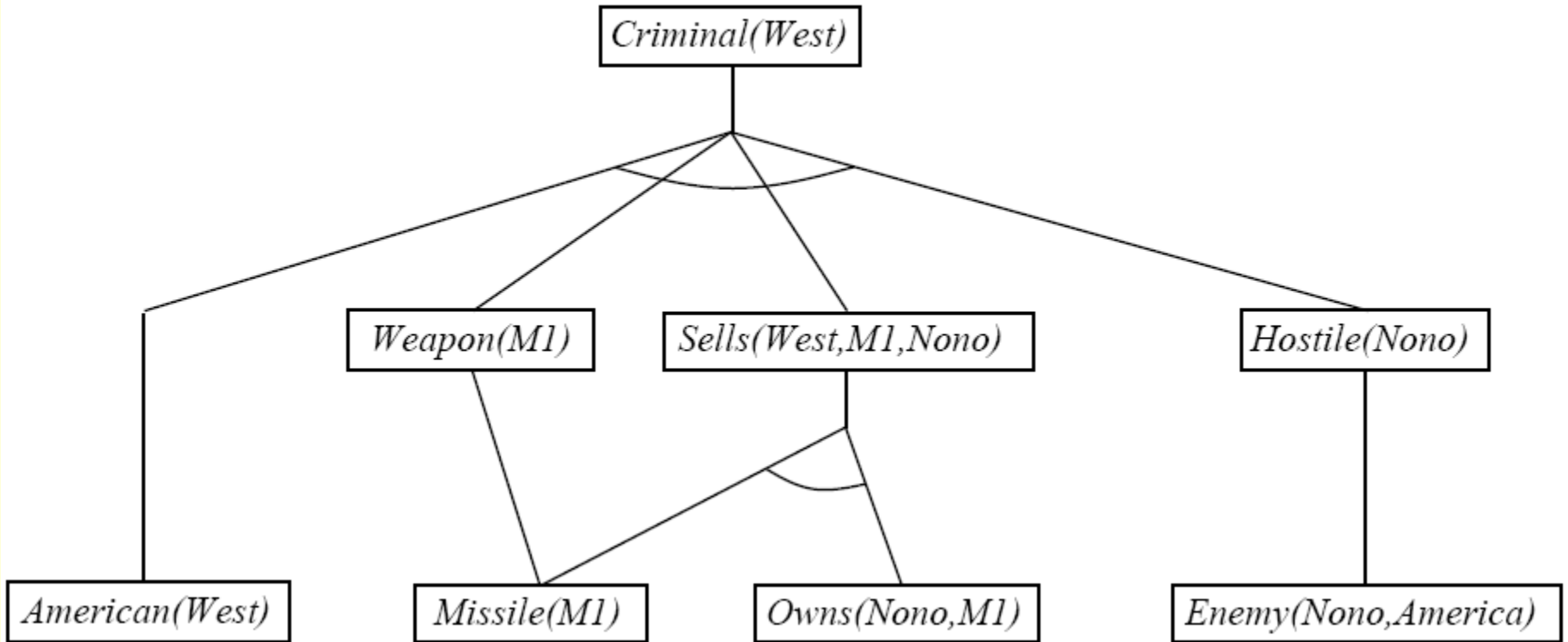
*Owns(Nono,M1)*

*Enemy(Nono,America)*

# FC: Example Knowledge Base



# FC: Example Knowledge Base



# Forward Chaining Algorithm

```
function FOL-FC-ASK( $KB, \alpha$ ) returns a substitution or false
  repeat until new is empty
     $new \leftarrow \{ \}$ 
    for each sentence  $r$  in  $KB$  do
       $(p_1 \wedge \dots \wedge p_n \Rightarrow q) \leftarrow \text{STANDARDIZE-APART}(r)$ 
      for each  $\theta$  such that  $(p_1 \wedge \dots \wedge p_n)\theta = (p'_1 \wedge \dots \wedge p'_n)\theta$ 
        for some  $p'_1, \dots, p'_n$  in  $KB$ 
           $q' \leftarrow \text{SUBST}(\theta, q)$ 
          if  $q'$  is not a renaming of a sentence already in  $KB$  or new then do
            add  $q'$  to new
             $\phi \leftarrow \text{UNIFY}(q', \alpha)$ 
            if  $\phi$  is not fail then return  $\phi$ 
    add new to  $KB$ 
  return false
```

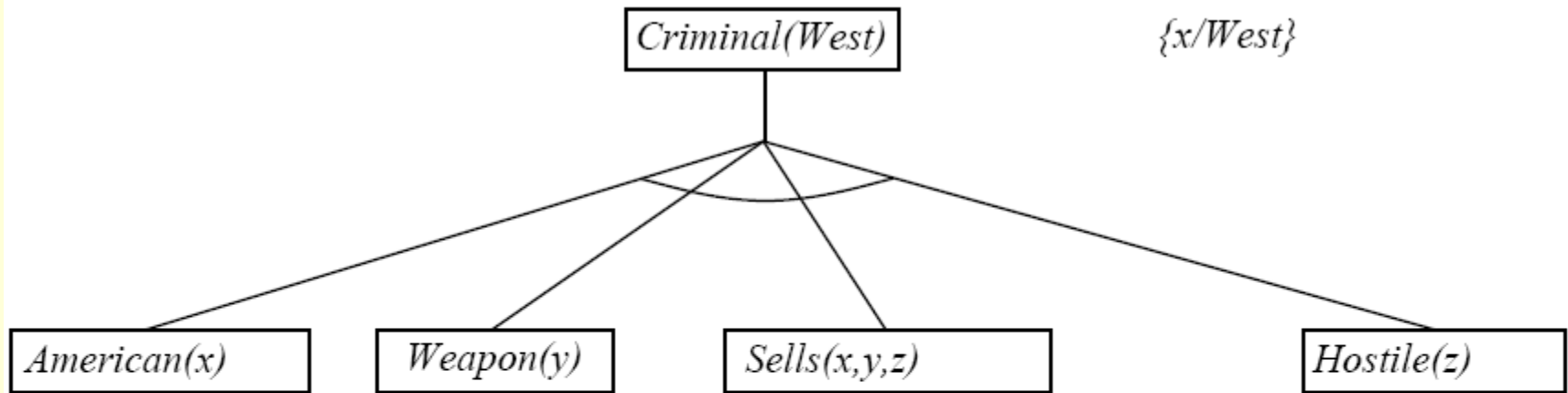
# Backward Chaining

- Consider the item to be proven a goal
- Find a rule whose head is the goal (and bindings)
- Apply bindings to the body, and prove these (subgoals) in turn
- If you prove all the subgoals, increasing the binding set as you go, you will prove the item.
- Logic Programming (cprolog, on CS)

# Backward Chaining Example

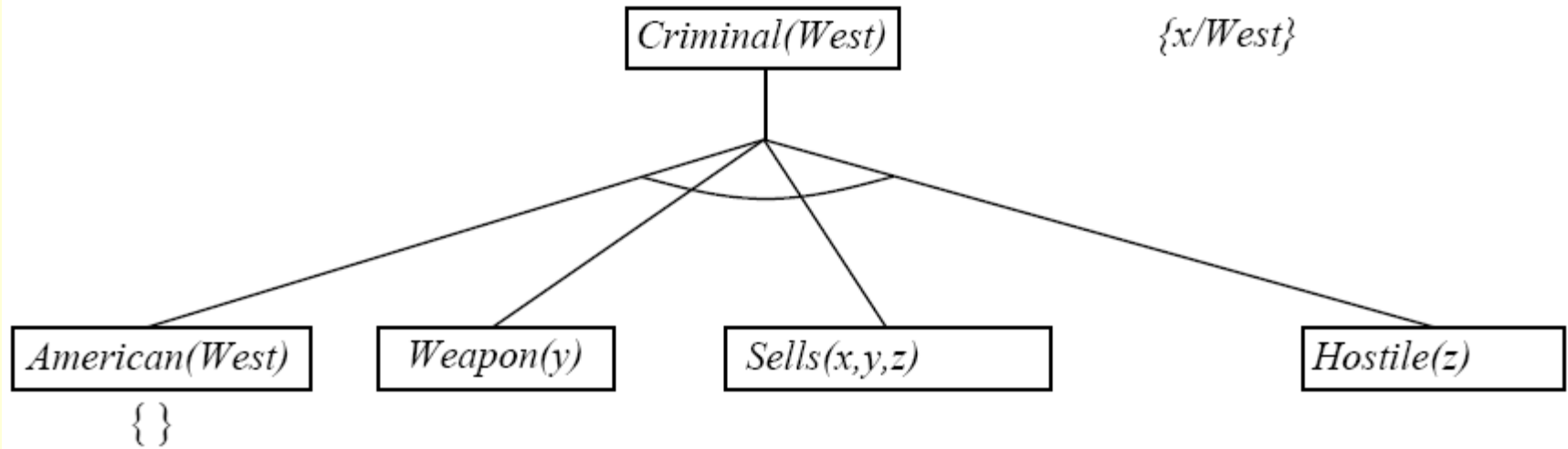
*Criminal(West)*

# Backward Chaining Example

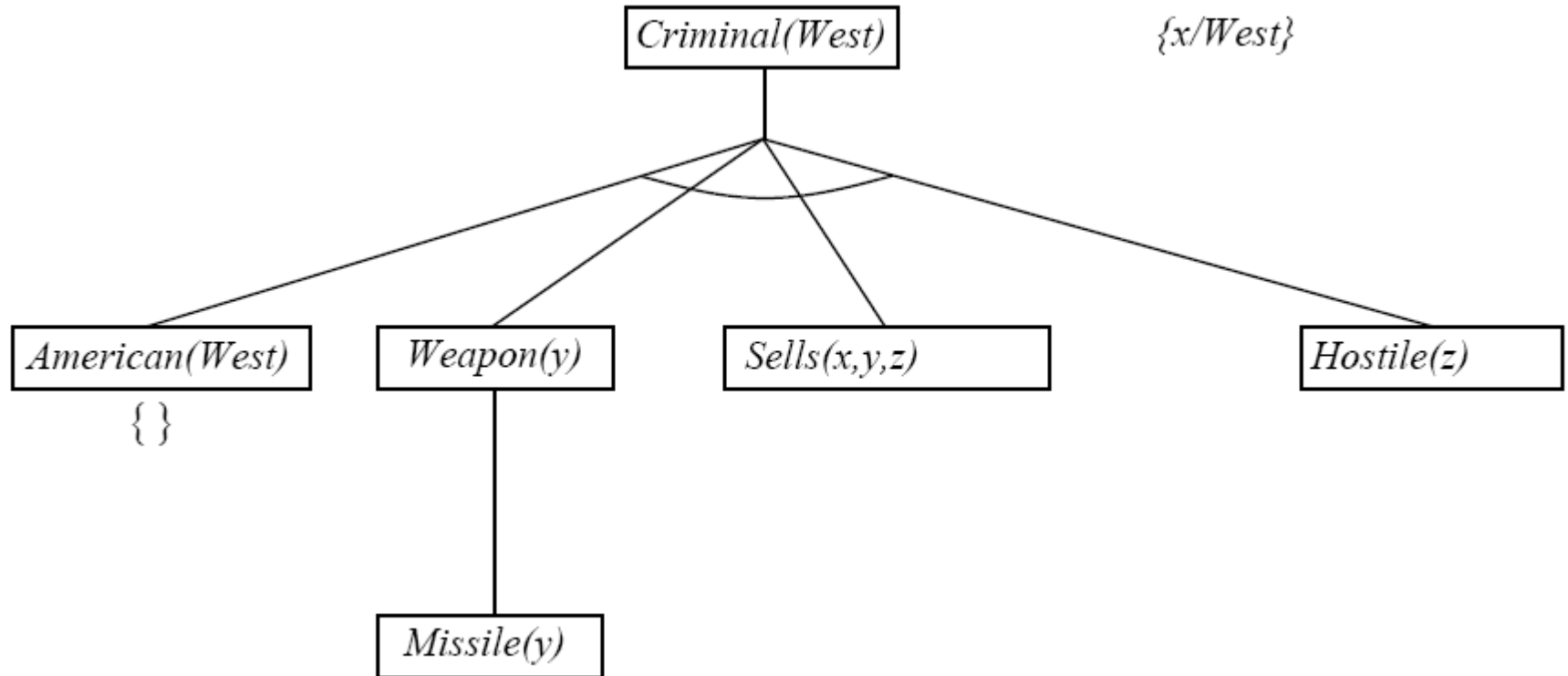




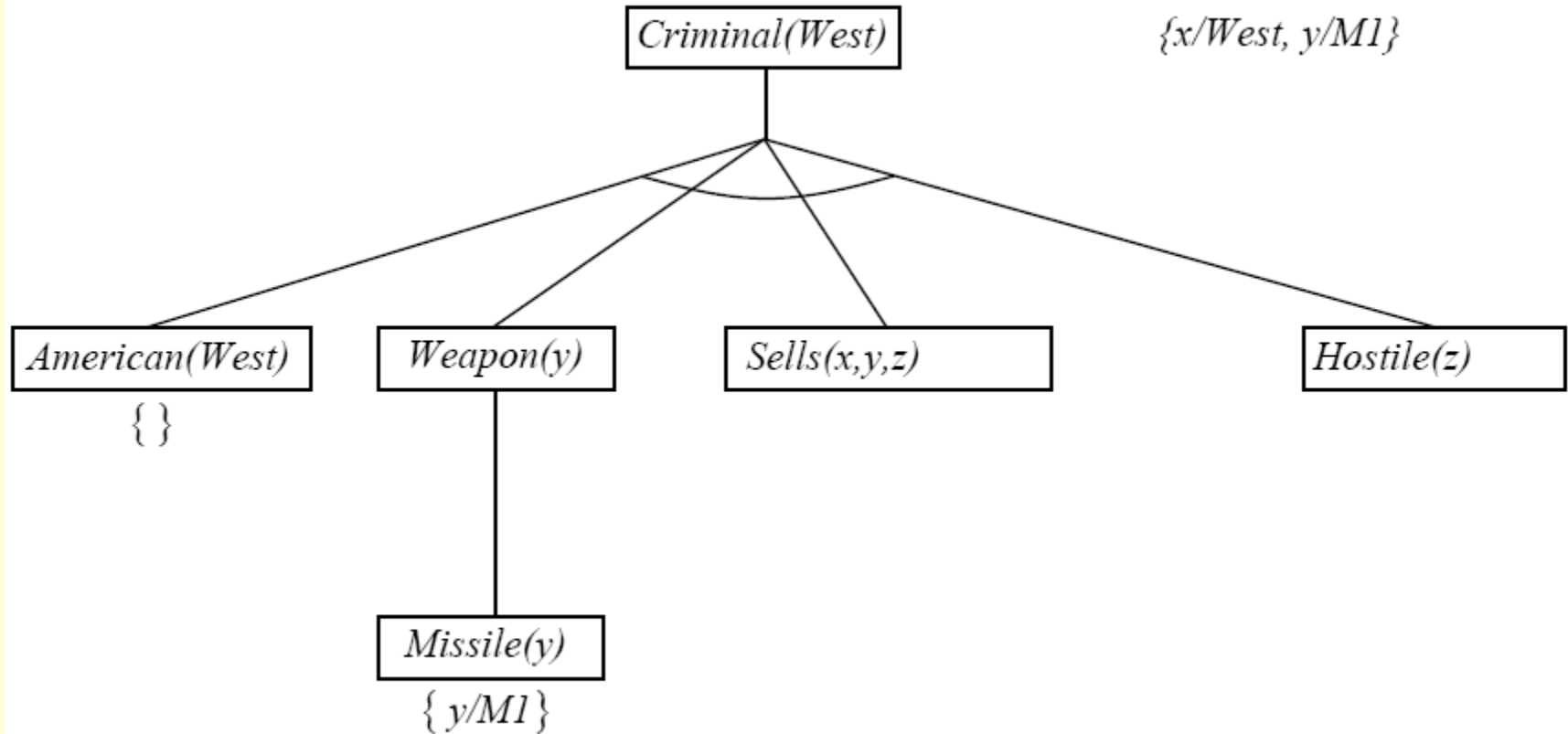
# Backward Chaining Example



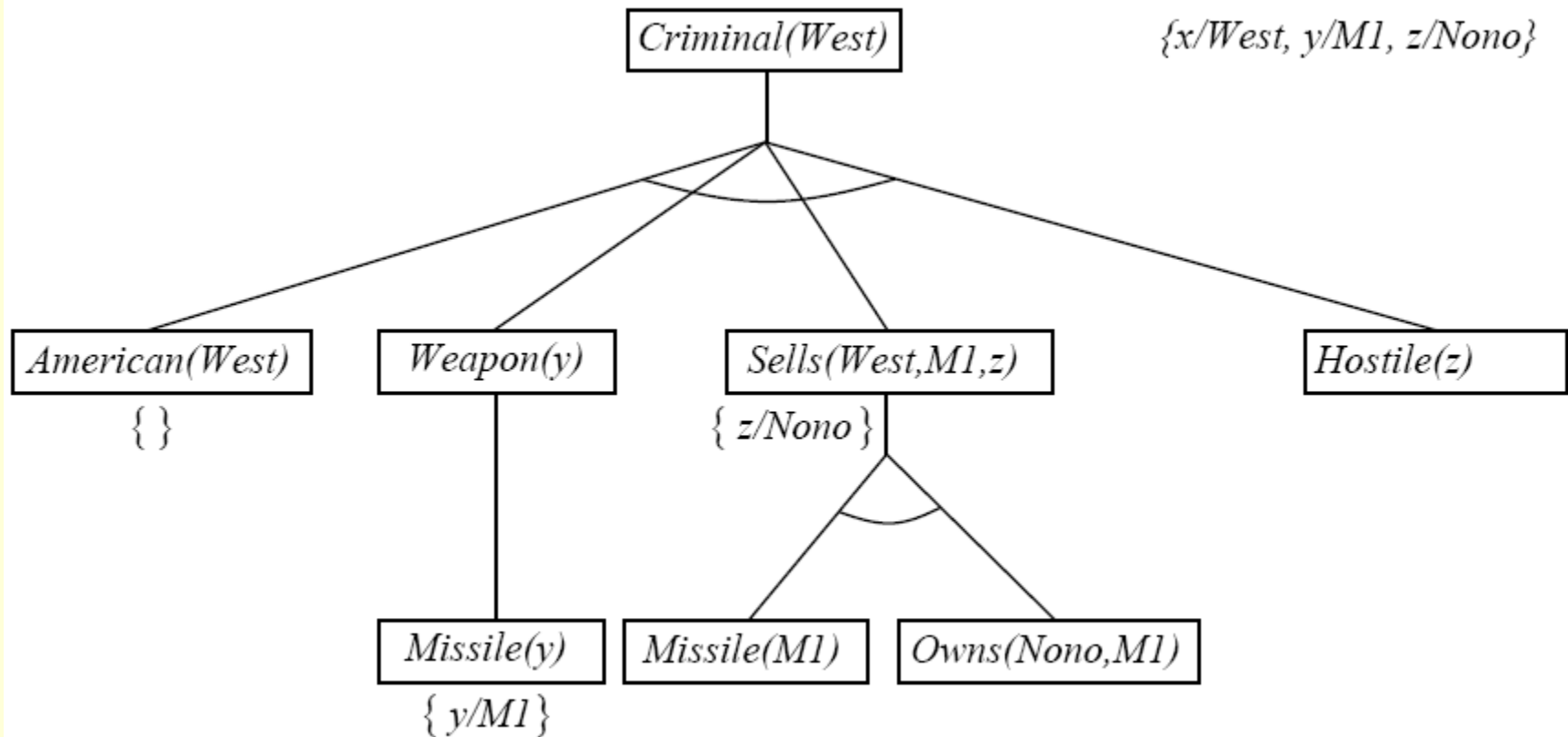
# Backward Chaining Example



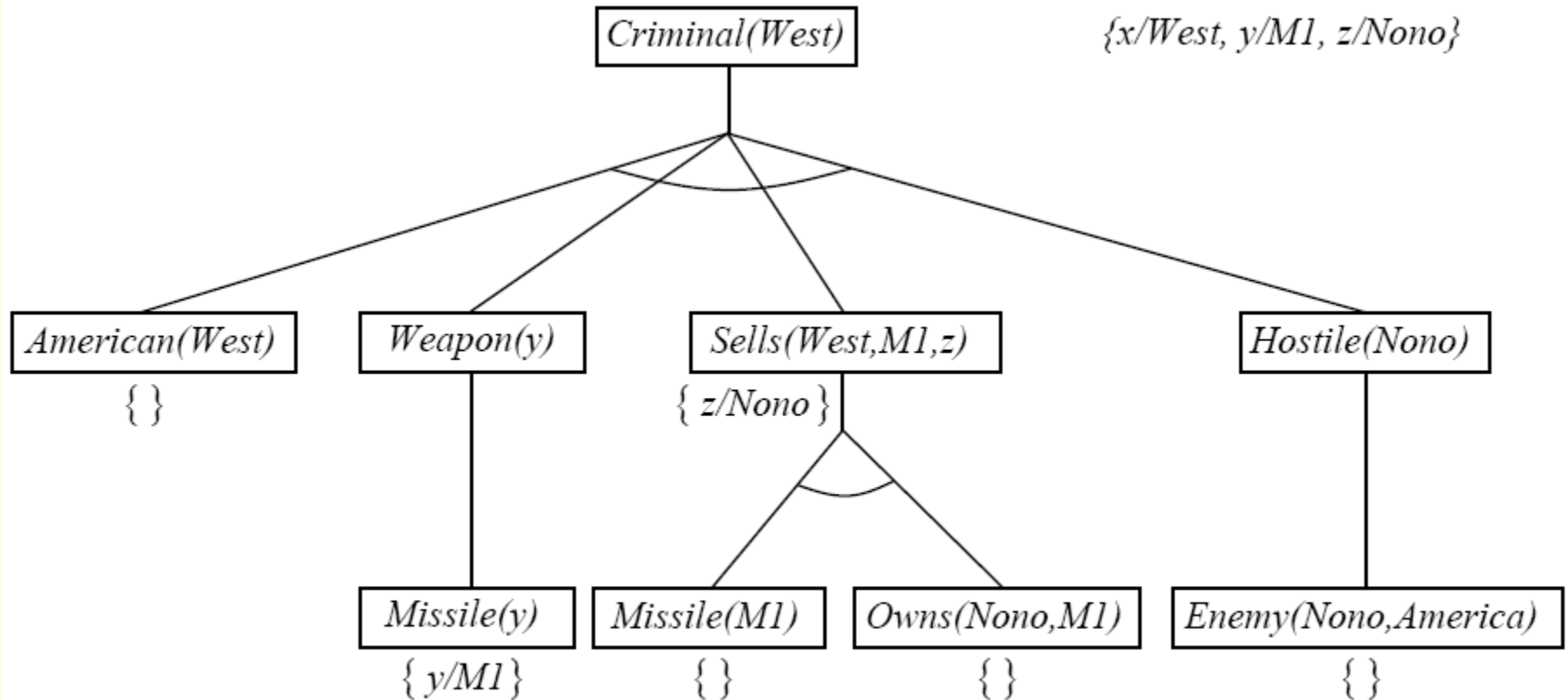
# Backward Chaining Example



# Backward Chaining Example



# Backward Chaining Example



# Backward Chaining Algorithm

```
function FOL-BC-ASK(KB, goals,  $\theta$ ) returns a set of substitutions
  inputs: KB, a knowledge base
            goals, a list of conjuncts forming a query
             $\theta$ , the current substitution, initially the empty substitution { }
  local variables: ans, a set of substitutions, initially empty

  if goals is empty then return { $\theta$ }
   $q' \leftarrow$  SUBST( $\theta$ , FIRST(goals))
  for each r in KB where STANDARDIZE-APART(r) = ( $p_1 \wedge \dots \wedge p_n \Rightarrow q$ )
    and  $\theta' \leftarrow$  UNIFY(q,  $q'$ ) succeeds
       $ans \leftarrow$  FOL-BC-ASK(KB, [ $p_1, \dots, p_n$  | REST(goals)], COMPOSE( $\theta'$ ,  $\theta$ ))  $\cup ans$ 
  return ans
```

# Properties of Backward Chaining

- Depth-first recursive proof search: space is linear in size of proof
- Incomplete due to infinite loops
  - Fix by checking current goal with every subgoal on the stack
- Inefficient due to repeated subgoals (both success and failure)
  - Fix using caching of previous results (extra space)
- Widely used without improvements for logic programming

# Resolution

- Convert everything to CNF (Conjunctive Normal Form)
- Resolve, with unification
- If resolution is successful, proof succeeds
- If there was a variable in the item to prove, return variable's value from unification bindings



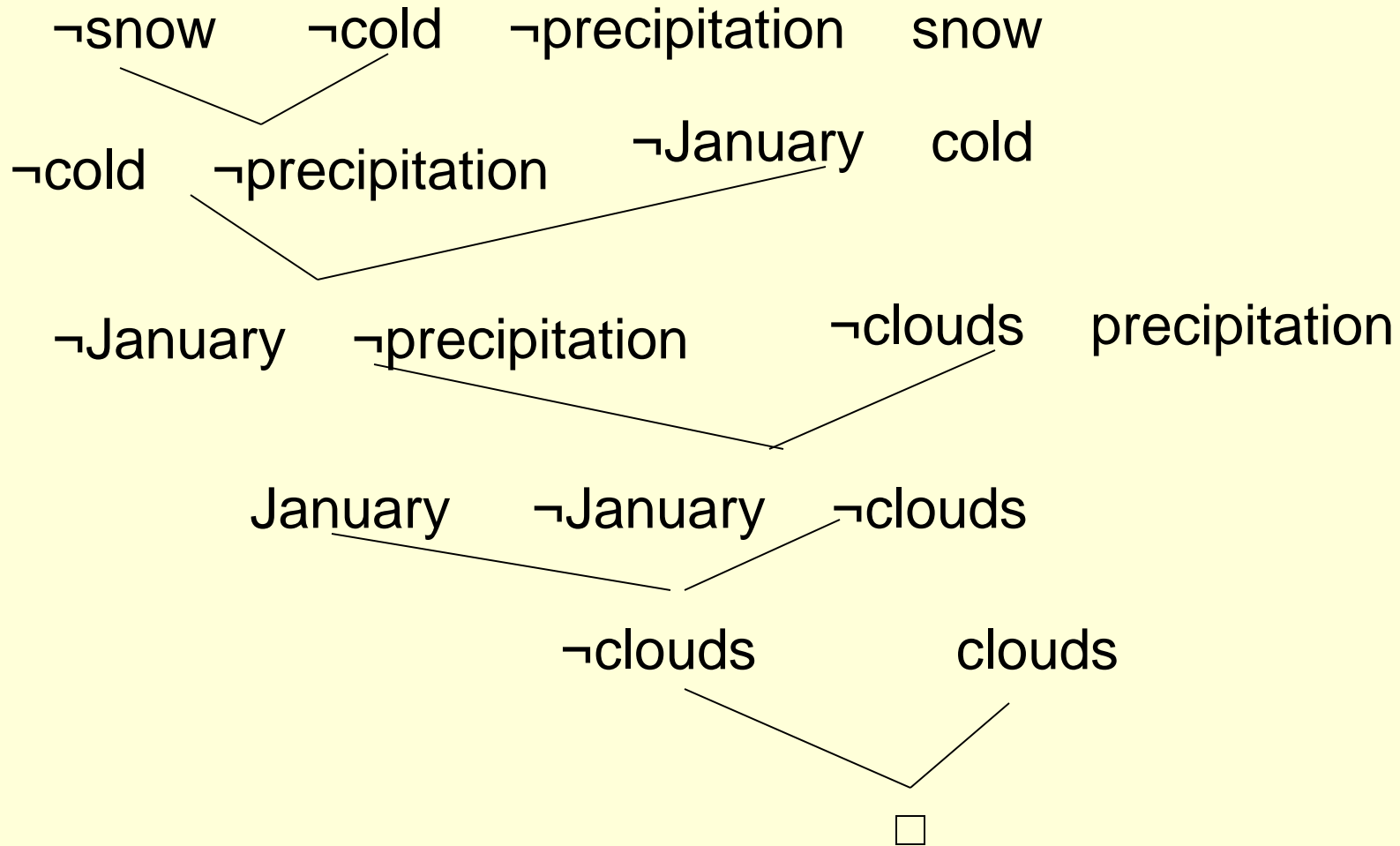
# Resolution (Review)

- Resolution allows a complete inference mechanism (search-based) using only one rule of inference
- Resolution rule:
  - Given:  $P_1 \quad P_2 \quad P_3 \quad \dots \quad P_n$ , and  $\neg P_1 \quad Q_1 \quad \dots \quad Q_m$
  - Conclude:  $P_2 \quad P_3 \quad \dots \quad P_n \quad Q_1 \quad \dots \quad Q_m$
  - Complementary literals  $P_1$  and  $\neg P_1$  "cancel out"
- To prove a proposition  $F$  by resolution,
  - Start with  $\neg F$
  - Resolve with a rule from the knowledge base (that contains  $F$ )
  - Repeat until all propositions have been eliminated
  - If this can be done, a contradiction has been derived and the original proposition  $F$  must be true.

# Propositional Resolution Example

- Rules
  - Cold and precipitation  $\rightarrow$  snow  
 $\neg$ cold  $\neg$ precipitation snow
  - January  $\rightarrow$  cold  
 $\neg$ January cold
  - Clouds  $\rightarrow$  precipitation  
 $\neg$ clouds precipitation
- Facts
  - January, clouds
- Prove
  - snow

# Propositional Resolution Example



# Resolution Theorem Proving (FOL)

- Convert everything to CNF
- Resolve, with unification
  - Save bindings as you go!
- If resolution is successful, proof succeeds
- If there was a variable in the item to prove, return variable's value from unification bindings

# Converting to CNF

1. Replace implication ( $A \rightarrow B$ ) by  $\neg A \vee B$
2. Move “inwards”
  - $\neg \forall x P(x)$  is equivalent to  $\exists x \neg P(x)$  & vice versa
3. Standardize variables
  - $\forall x P(x) \wedge \exists x Q(x)$  becomes  $\forall x P(x) \wedge \exists y Q(y)$
4. Skolemize
  - $\forall x P(x)$  becomes  $P(A)$
5. Drop universal quantifiers
  - Since all quantifiers are now  $\exists$ , we don't need them
6. Distributive Law

# Convert to FOPL, then CNF

1. John likes all kinds of food
2. Apples are food.
3. Chicken is food.
4. Anything that anyone eats and isn't killed by is food.
5. Bill eats peanuts and is still alive.
6. Sue eats everything Bill eats.

# Prove Using Resolution

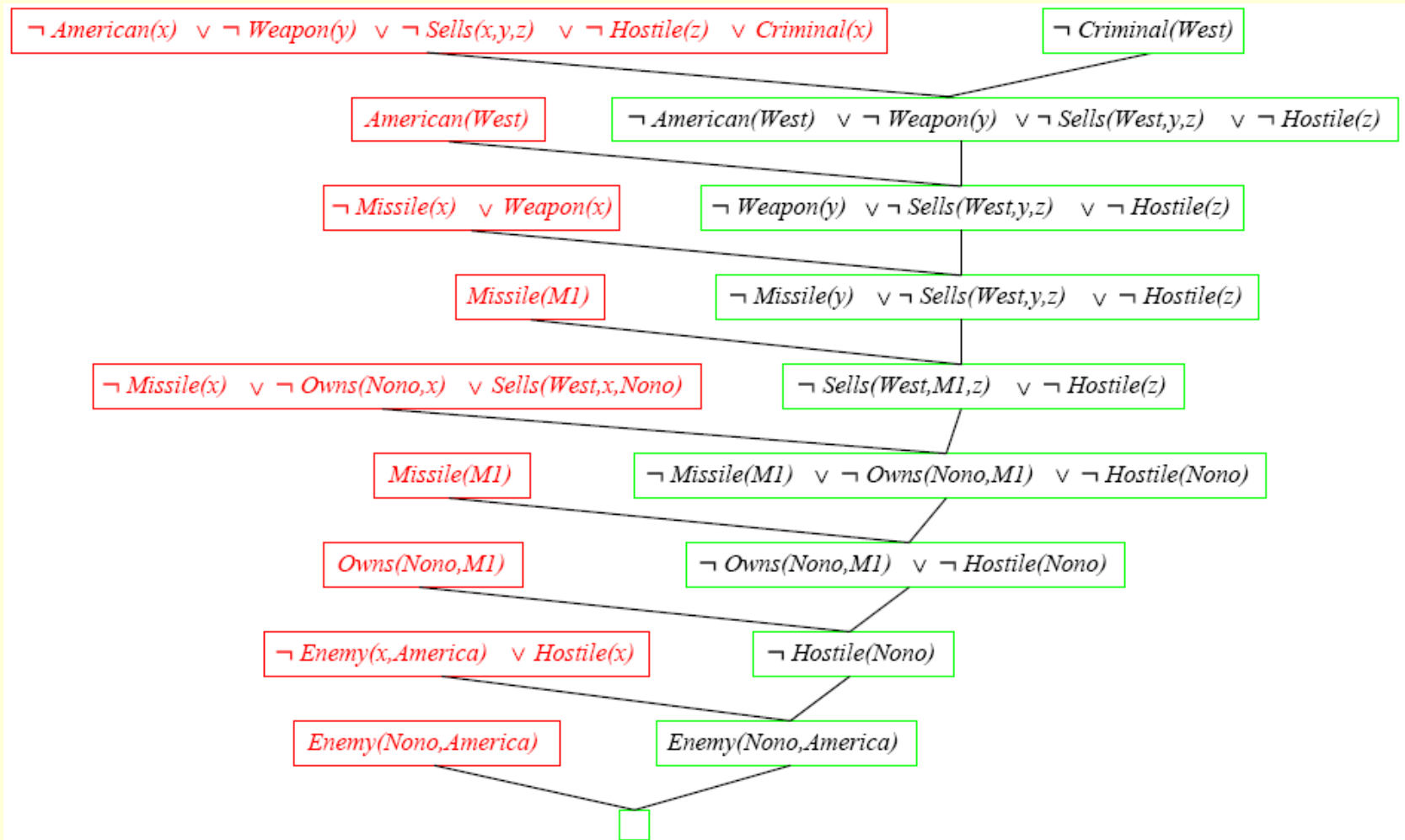
1. John likes peanuts.
2. Sue eats peanuts.
3. Sue eats apples.
4. What does Sue eat?
  - Translate to Sue eats X
  - Result is a valid binding for X in the proof

# Another Example

- Steve only likes easy courses
- Science courses are hard
- All the courses in the basket weaving department are easy
- BK301 is a basket weaving course
- What course would Steve like?



# Another Resolution Example



# Final Thoughts on Resolution

- Resolution is complete. If you don't want to take this on faith, study pp. 300-303
- Strategies (heuristics) for efficient resolution include
  - Unit preference. If a clause has only one literal, use it first.
  - Set of support. Identify "useful" rules and ignore the rest. (p. 305)
  - Input resolution. Intermediately generated sentences can only be combined with original inputs or original rules. (We used this strategy in our examples).
  - Subsumption. Prune unnecessary facts from the database.

# CMSC 310 Artificial Intelligence

## Probabilistic Reasoning and Bayesian Belief Networks

Probabilities, Random Variables, Probability Distribution, Conditional Probability, Joint Distributions, Bayes Theorem

### 1. Probability of an Event

Consider an experiment that may have different outcomes. We are interested to know what is the probability of a particular set of outcomes.

Let sample space  $S$  be the set of all possible outcomes

Let Event  $A$  be any subset of  $S$

**Definition 1: probability(A)** = (number of outcomes in  $A$ ) / (total number of outcomes)

$$P(A) = |A| / |S|$$

i.e. the probability of  $A$  is equal to the number of outcomes of interest divided by the number of all possible outcomes.

$P(A)$  is called prior (unconditional) probability of  $A$

$P(\sim A)$  is the probability event  $A$  not to take place.

**Example 1:** the probability to pick a spade card out of a deck of 52 cards is  $13/52 = 1/4$

The probability to pick an Ace out of a deck of 52 cards is  $4/52 = 1/13$

**Probability Axioms:**

$$(1) 0 \leq P(A) \leq 1$$

$$(2) P(A) = 1 - P(\sim A)$$

$$(3) P(A \vee B) = P(A) + P(B) - P(A \& B)$$

$P(A \vee B)$  means the probability of either  $A$  or  $B$  or both to be true

$P(A \& B)$  means the probability of both  $A$  and  $B$  to be true.

**Example 2:  $P(\sim A)$**  – The probability to pick a card that is not a spade out of a deck of 52 cards is  
 $1 - 1/4 = 3/4$

**Example 3:  $P(A \vee B)$**  – The probability to pick a card that is either a spade or an Ace is  
 $1/4 + 1/13 - 1/4 * 1/13 = 16/52 = 4/13$

Another way to obtain the same result: There are 13 spade cards and 3 additional Ace cards in the set of desired outcomes. The total number of cards is 52, thus the probability is  $16/52$ .

**Example 4:  $P(A \& B)$**  – The probability to pick the spade Ace is  $1/52$

## 2. Random Variables and Probability Distributions

To handle more conveniently the outcomes, we can treat them as values of so called **random variables**. For example “spade” is one possible value of the variable Suit, “clubs” is another possible value. In the card example, all values of the variable Suit are equally probable. This is not always so however. We may be interested in the probabilities of each separate value.

**The set of the probabilities of each value is called probability distribution of the random variable.**

Let  $X$  be a random variable with a domain  $\langle x_1, x_2, \dots, x_n \rangle$

The probability distribution of  $X$  is denoted by  $\mathbf{P}(X) = \langle P(X = x_1), P(X = x_2), \dots, P(X = x_n) \rangle$

Note that  $P(X = x_1) + P(X = x_2) + \dots + P(X = x_n) = 1$

**Example 5:** Let Weather be a random variable with values  $\langle \text{sunny, cloudy, rainy, snowy} \rangle$

Assume that records for some town show that in a year 100 days are rainy, 50 days are snowy, 120 days are cloudy (but without snow or rain) and 95 days are sunny.

i.e.  $P(\text{Weather} = \text{sunny}) = 95/365 = 0.26$   
 $P(\text{Weather} = \text{cloudy}) = 120/365 = 0.33$   
 $P(\text{Weather} = \text{rainy}) = 100/365 = 0.27$   
 $P(\text{Weather} = \text{snowy}) = 50/365 = 0.14$

Thus  $\mathbf{P}(\text{Weather}) = \langle 0.26, 0.33, 0.27, 0.14 \rangle$  is the probability distribution of the random variable Weather.

## 3. Joint Distributions

The following example is used to illustrate conditional probabilities and joint distributions

**Example 6:** Consider a sample  $S$  of 1000 individuals age 25 – 30. Assume that 600 individuals come from high-income families, 570 of those with high income have college education and 100 individuals with low income have college education.

The following table illustrates the example:

		C	$\sim C$	
		College ed.	Not college ed.	
H	High income	570	30	600
$\sim H$	Low income	100	300	400
		670	330	1000

Let  $H$  be the subset of  $S$  of individuals coming from high-income families,  $|H| = 600$

Let  $C$  be the subset of  $S$  of individuals that have college education,  $|C| = 670$

The prior probabilities of H, ~H, C and ~C are:

$$P(H) = 600 / 1000 = 0.6 \text{ (60\%)} \quad P(\sim H) = 400 / 1000 = 0.4 \text{ (40\%)}$$

$$P(C) = 670 / 1000 = 0.67 \text{ (67\%)} \quad P(\sim C) = 330 / 1000 = 0.33 \text{ (33\%)}$$

We can compute also  $P(H \& C)$ ,  $P(H \& \sim C)$ ,  $P(\sim H \& C)$ ,  $P(\sim H \& \sim C)$

$P(H \& C) = |H \& C| / |S| = 570/1000 = 0.57 \text{ (57\%)}$  - the probability of a randomly selected individual in S to be of high-income family and to have college education.

$P(H \& \sim C) = |H \& \sim C| / |S| = 30/1000 = 0.03 \text{ (3\%)}$  - the probability of a randomly selected individual in S to be of high-income family and not to have college education.

$P(\sim H \& C) = |\sim H \& C| / |S| = 100/1000 = 0.1 \text{ (10\%)}$  - the probability of a randomly selected individual in S to be of low-income family and to have college education.

$P(\sim H \& \sim C) = |\sim H \& \sim C| / |S| = 300/1000 = 0.3 \text{ (30\%)}$  - the probability of a randomly selected individual in S to be of low-income family and not to have college education.

Thus we come to the following table:

		C	~C	
		College ed.	Not college ed.	
H	High income	0.57	0.03	0.6
~H	Low income	0.10	0.30	0.4
		0.67	0.33	1

Here we will treat C and H as random variables with values “yes” and “no”. The values in the table represent the **joint distribution** of C and H, forexample

$$P(C = \text{yes}, H = \text{yes}) = 0.57$$

Formally, joint distribution is defined as follows:

**Definition 2:** Let  $X_1, X_2, \dots, X_n$  be a set of random variables each with a range of specific values.

$P(X_1, X_2, \dots, X_n)$  is called **joint distribution** of the variables  $X_1, X_2, \dots, X_n$  and it is defined by a n-dimensional table, where each cell corresponds to one particular assignment of values to the variables  $X_1, X_2, \dots, X_n$

Each cell in the table corresponds to an **atomic event** – described by a particular assignment of values to the variables.

Since the atomic events are mutually exclusive, their conjunction is necessarily false.

Since they are collectively exhaustive, the disjunction is necessarily true.

So by axioms (2) and (3) the sum of all entries in the table is 1

Given a joint distribution table we can compute prior probabilities:

$$P(H) = P(H \& C) + P(H \& \sim C) = 0.57 + 0.03 = 0.6$$

Given a joint distribution table we can compute conditional probabilities, discussed in the next section.

#### 4. Conditional Probabilities

We may ask: what is the probability of an individual in  $S$  to have a college education given that he/she comes from a high income family?

In this case we consider only those individuals that come from high income families. Their number is 600. The number of individuals with college education within the group of high-family income is 570. Thus the probability to have college education given high-income family is  $570/600 = 0.95$ .

This type of probability is called **conditional probability**

The probability of event  $B$  given event  $A$  is denoted as  $P(B|A)$ , read “P of B given A”

$$\text{In our example, } P(C|H) = \frac{|C \& H|}{|H|}$$

We will represent  $P(C|H)$  by  $P(C\&H)$  and  $P(H)$

$$P(C|H) = \frac{|C \& H|}{|H|} = \frac{\frac{|C \& H|}{|S|}}{\frac{|H|}{|S|}} = \frac{P(C\&H)}{P(H)}$$

Therefore

$$P(C|H) = P(C\&H) / P(H)$$

**Definition 3:** The conditional probability of an event  $B$  to occur given that event  $A$  has occurred is

$$P(B|A) = P(B\&A) / P(A)$$

$P(B|A)$  is known also as **posterior probability** of  $B$

$P(B \& A)$  is an element of the joint distribution of the random variables  $A$  and  $B$ .

In our example,  $P(C\&H) = P(C = \text{yes}, H = \text{yes})$ . Thus given the joint distribution  $\mathbf{P}(H, C)$ , we can compute the prior probability  $P(H)$ ,  $P(\sim H)$ ,  $P(C)$ ,  $P(\sim C)$  and then the conditional probability  $P(C|H)$ ,  $P(C|\sim H)$ ,  $P(H|C)$ ,  $P(H|\sim C)$ .

## Independent events

Some events are not related, for example each outcome in a sequence of coin flips is independent on the previous outcome.

**Definition 4:** Two events **A and B are independent** if  $P(A|B) = P(A)$ , and  $P(B|A) = P(B)$ .

**Theorem:** A and B are independent if and only if  $P(A \& B) = P(A)*P(B)$

The proof follows directly from Definition 3 and Definition 4.

**Another definition:** X and Y are conditionally independent iff  $P(X|Y \& Z) = P(X|Z)$

## 5. Bayes' Theorem

From Definition 3 we have

$$\begin{aligned} P(A\&B) &= P(A|B)*P(B) \\ P(B\&A) &= P(B|A)*P(A) \end{aligned}$$

However,  $P(A\&B) = P(B\&A)$

Therefore

$$P(B|A)*P(A) = P(A|B)*P(B)$$

$$P(B|A) = \frac{P(A|B) * P(B)}{P(A)}$$

This is the Bayes' formula for conditional probabilities, known also as Bayes' theorem

### 5.1. More than 2 variables

Bayes' theorem can represent conditional probability for more than two variables:

$$P(X|Y_1\&Y_2 \& \dots\& Y_n) = P(Y_1 \& Y_2 \& \dots \& Y_n | X) * P(X) / P(Y_1 \& Y_2 \& \dots \& Y_n)$$

Think of X as being a hypothesis, and  $Y_1, Y_2, \dots, Y_n$  as being n pieces of evidence for the hypothesis. When  $Y_1, Y_2, \dots, Y_n$  are independent on each other, the formula takes the form:

$$P(X|Y_1\&Y_2 \& \dots\& Y_n) = \frac{P(Y_1|X)*P(Y_2|X)*\dots*P(Y_n | X) * P(X)}{P(Y_1)*P(Y_2)*\dots*P(Y_n)}$$

In case of several related events, the Bayes' formula is used in the following form:

$$P(X_1 \& X_2 \& \dots \& X_n) = P(X_1) * P(X_2|X_1) * P(X_3 | X_2 \& X_1) \dots P(X_n | X_{n-1} \& \dots X_1)$$

## 5.2. Normalization

Consider the probability of malaria given headache

$$P(M|H) = P(H | M) * P(M) / P(H)$$

It may be more difficult to compute P(H) than P(H|M) and P(H | ~M).

**We can represent P(H) through P(H|M) and P(H | ~M).**

We have:

$$P(M|H) = P(H | M) * P(M) / P(H)$$

$$P(\sim M|H) = P(H | \sim M) * P(\sim M) / P(H)$$

Adding these equations we obtain

$$P(M|H) + P(\sim M|H) = ( P(H | M) * P(M) + P(H | \sim M) * P(\sim M) ) / P(H)$$

For the left side we know that  $P(M|H) + P(\sim M|H) = 1$

So we have

$$1 = ( P(H | M) * P(M) + P(H | \sim M) * P(\sim M) ) / P(H)$$

Multiply both sides by P(H):

$$P(H) = P(H | M) * P(M) + P(H | \sim M) * P(\sim M)$$

Replacing in the Bayes' Theorem P(H) with the right side above, we get:

$$P(M|H) = \frac{P(H | M) * P(M)}{P(H | M) * P(M) + P(H | \sim M) * P(\sim M)}$$

This process is called normalization because it resembles the normalization process for functions – multiplying a function by a chosen constant so that its values stay within a specified range.

## 5.3. Relative Likelihood of two events

Given that you have a headache, is it more likely that you have flu rather than plague?

$$P(\text{plague}|\text{headache}) = P(\text{headache} | \text{plague}) * P(\text{plague}) / P(\text{headache})$$

$$P(\text{flu} | \text{headache}) = P(\text{headache} | \text{flu}) * P(\text{flu}) / P(\text{headache})$$

The ratio

$$\frac{P(\text{plague}|\text{headache})}{P(\text{flu} | \text{headache})} = \frac{P(\text{headache} | \text{plague}) * P(\text{plague})}{P(\text{headache} | \text{flu}) * P(\text{flu})}$$

is called relative likelihood of having plague vs having flu given headache. It can be computed without knowing P(headache).



In general, the relative likelihood of two events B and C given A is computed as follows

$$\frac{P(B | A)}{P(C | A)} = \frac{P(A | B) * P(B)}{P(A | C) * P(C)}$$

### 5.4. Example: The Monty Hall game

You are about to choose your winning in a game show. There are three doors behind one of which is a red Porsche and other two, goats. You will get whatever is behind the door you choose. You pick a door, say A. At this point the game show host opens one of the other two doors, which he knows to contain a goat, for example B and asks if you would now like to revise your choice to C. The question is: Should you? (Assuming you want the car and not the goat.)

Let  $P(PA)$ ,  $P(PB)$ , and  $P(PC)$  be the probabilities of the Porsche being behind door A, door B and door C respectively. We assume that the car is randomly placed, so

$$P(PA) = P(PB) = P(PC) = 1/3$$

Let O be the event that Monty Hall opens door B.

The Monty Hall Problem can be restated as follows: is  $P(PA | O) = P(PC | O)$

By the Bayes' Theorem we have:

$$P(PA | O) = \frac{P(O | PA) * P(PA)}{P(O)}$$

$$P(PC | O) = \frac{P(O | PC) * P(PC)}{P(O)}$$

We have to compute  $P(O)$ ,  $P(O|PA)$  and  $P(O|PC)$

$P(O | PA) = 1/2$  , if the car is behind A, Monty Hall can open either B or C

$P(O | PB) = 0$  , if the car is behind B, Monty Hall will not open B

$P(O | PC) = 1$  , if the car is behind C, Monty Hall can only open door B

$P(O) = P(O|PA) * P(PA) + P(O|PB) * P(PB) + P(O|PC) * P(PC)$  (see section 5.2. Normalization)

$$P(O) = 1/3 * ( 1/2 + 0 + 1) = 1/2$$

Therefore we obtain:

$$P(PA | O) = ( 1 / 2 * 1 / 3 ) / ( 1 / 2 ) = 1/3$$

$$P(PC | O) = ( 1 * 1/3 ) / ( 1 / 2 ) = 2/3$$

So, if you switch to door C, you double your chance to win the Porsche.

## 5.5. Useful expressions

$$P(A|B) = \frac{P(A \& B)}{P(B)}$$

$$P(A|B) = \frac{P(A \& B)}{P(A \& B) + P(\sim A \& B)}$$

$$P(A | B) = \frac{P(B|A) * P(A)}{P(B)}$$

$$P(A | B) = \frac{P(B|A) * P(A)}{P(B|A)*P(A) + P(B|\sim A) * P(\sim A)}$$

## 6. Simple Bayesian Concept Learning

The Bayes' theorem can be used to solve the following problem:

Determine the most probable hypothesis out of n possible hypotheses  $H_1, H_2, \dots, H_n$ , given a set of evidence E. For each  $H_i$  we can compute

$$P(H_i | E) = \frac{P(E|H_i) * P(H_i)}{P(E)}$$

and take the hypothesis  $H_k$  for which  $P(H_k | E)$  has the greatest value.

This is a maximization problem – we are not looking for the particular value of each  $P(H_i | E)$ , we are looking the hypothesis for which the posterior probability is maximum. Hence we can simplify the expression to be computed based on the following considerations:

a) The evidence is not dependent on the hypotheses, so we can remove  $P(E)$  :

$$P(H_i | E) = P(E|H_i) * P(H_i)$$

b) Assuming that all hypotheses are equally likely (same prior probability), we can remove the prior probability

$$P(H_i | E) = P(E|H_i)$$

We choose the hypothesis for which the value of  $P(E|H_i)$  is highest.

$P(E|H_i)$  is known as the likelihood of the evidence E given the hypothesis  $H_i$ .

### Additional Reading:

[http://www.dartmouth.edu/~chance/teaching\\_aids/books\\_articles/probability\\_book/Chapter4.pdf](http://www.dartmouth.edu/~chance/teaching_aids/books_articles/probability_book/Chapter4.pdf)

## Utility function

- **Utility function (denoted U)**
  - Quantifies how we “value” outcomes, i.e., it reflects our preferences
  - Can be also applied to “value” outcomes other than money and gains (e.g. utility of a patient being healthy, or ill)
- **Decision making:**
  - uses expected utilities (denoted EU)

$$EU(X) = \sum_{x \in X} P(X=x)U(X=x)$$

$U(X=x)$  the utility of outcome x

### **Important !!!**

- Under some conditions on preferences **we can always design the utility function that fits our preferences**

## Utility theory

- Defines axioms on preferences that involve uncertainty and ways to manipulate them.
- Uncertainty is modeled through **lotteries**
  - **Lottery:**

$$[p : A; (1 - p) : C]$$
    - Outcome A with probability p
    - Outcome C with probability (1-p)
- The following six constraints are known as the axioms of utility theory. The axioms are the most obvious semantic constraints on preferences with lotteries.
  - indifferent (equally preferable)

## Axioms of the utility theory

- **Orderability:** Given any two states, the a rational agent prefers one of them, else the two as equally preferable.
 
$$(A \preceq B) \quad (B \preceq A) \quad (A \sim B)$$
- **Transitivity:** Given any three states, if an agent prefers  $A$  to  $B$  and prefers  $B$  to  $C$ , agent must prefer  $A$  to  $C$ .
 
$$(A \preceq B) \quad (B \preceq C) \quad (A \preceq C)$$
- **Continuity:** If some state  $B$  is between  $A$  and  $C$  in preference, then there is a  $p$  for which the rational agent will be indifferent between state  $B$  and the lottery in which  $A$  comes with probability  $p$ ,  $C$  with probability  $(1-p)$ .

$$(A \preceq B \preceq C) \quad , \quad p [p : A; (1 - p) : C] \sim B$$

## Axioms of the utility theory

- **Substitutability:** If an agent is indifferent between two lotteries,  $A$  and  $B$ , then there is a more complex lottery in which  $A$  can be substituted with  $B$ .

$$(A \sim B) \quad [p : A; (1 - p) : C] \sim [p : B; (1 - p) : C]$$

- **Monotonicity:** If an agent prefers  $A$  to  $B$ , then the agent must prefer the lottery in which  $A$  occurs with a higher probability

$$(A \succ B) \quad (p > q) \quad [p : A; (1 - p) : B] \succ [q : A; (1 - q) : B]$$

- **Decomposability:** Compound lotteries can be reduced to simpler lotteries using the laws of probability.

$$[p : A; (1 - p) : [q : B; (1 - q) : C]]$$

$$[p : A; (1 - p)q : B; (1 - p)(1 - q) : C]$$

## Utility theory

**If the agent obeys the axioms of the utility theory, then**

1. there exists a real valued function  $U$  such that:

$$U(A) > U(B) \quad A \succ B$$

$$U(A) = U(B) \quad A \sim B$$

3. The utility of the lottery is the expected utility, that is the sum of utilities of outcomes weighted by their probability

$$U[p : A; (1 - p) : B] = pU(A) + (1 - p)U(B)$$

5. Rational agent makes the decisions in the presence of uncertainty by maximizing its expected utility

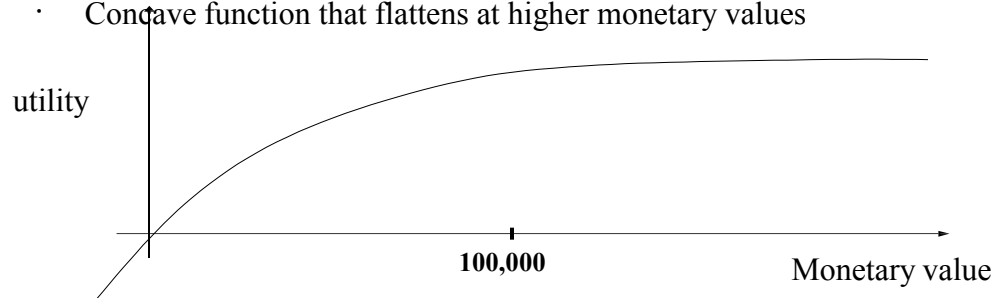
## Utility functions

We can design a utility function that fits our preferences if they satisfy the axioms of utility theory.

- **But how to design the utility function for monetary values so that they incorporate the risk?**
- **What is the relation between utility function and monetary values?**
- Assume we loose or gain \$1000.
  - Typically this difference is more significant for lower values (around \$100 -1000) than for higher values (~ \$1,000,000)
- What is the relation between utilities and monetary value for a typical person?

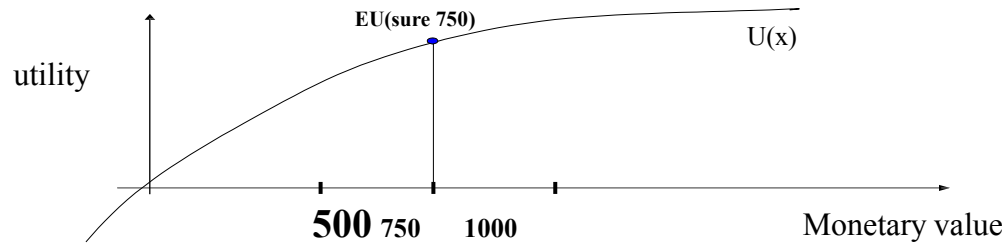
## Utility functions

- What is the relation between utilities and monetary value for a typical person?
- Concave function that flattens at higher monetary values



## Utility functions

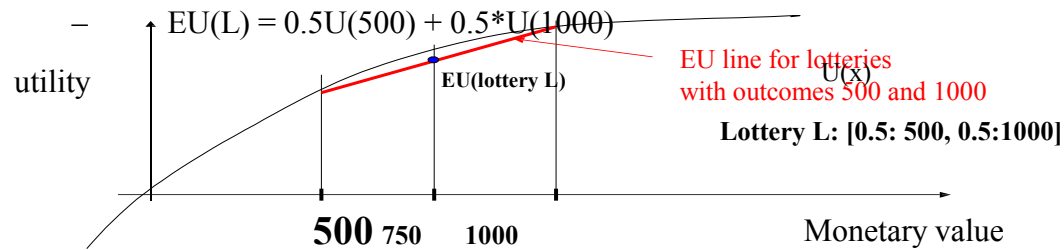
- Expected utility of a sure outcome of 750 is 750



## Utility functions

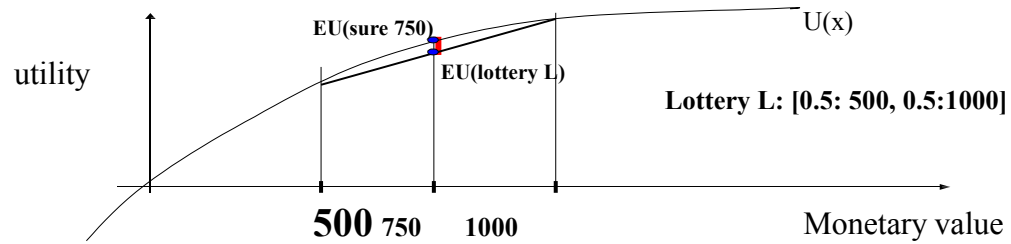
Assume a lottery L [0.5: 500, 0.5:1000]

- Expected value of the lottery = 750
- Expected utility of the lottery  $EU(L)$  is different:



## Utility functions

- Expected utility of the lottery  $EU(\text{lottery L}) < EU(\text{sure } 750)$



- Risk aversion – a bonus is required for undertaking the risk



# Hidden Markov model

A **hidden Markov model (HMM)** is a [statistical Markov model](#) in which the system being modeled is assumed to be a [Markov process](#) with unobserved (*hidden*) states. An HMM can be presented as the simplest [dynamic Bayesian network](#). The mathematics behind the HMM were developed by [L. E. Baum](#) and coworkers. It is closely related to an earlier work on the optimal nonlinear [filtering problem](#) by Ruslan L. Stratonovich who was the first to describe the [forward-backward procedure](#).

In simpler [Markov models](#) (like a [Markov chain](#)), the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters. In a *hidden* Markov model, the state is not directly visible, but the output, dependent on the state, is visible. Each state has a probability distribution over the possible output tokens. Therefore, the sequence of tokens generated by an HMM gives some information about the sequence of states. The adjective 'hidden' refers to the state sequence through which the model passes, not to the parameters of the model; the model is still referred to as a 'hidden' Markov model even if these parameters are known exactly.

Hidden Markov models are especially known for their application in [temporal](#) pattern recognition such as [speech](#), [handwriting](#), [gesture recognition](#), [part-of-speech tagging](#), musical score following, [partial discharges](#) and [bioinformatics](#).

A hidden Markov model can be considered a generalization of a [mixture model](#) where the hidden variables (or [latent variables](#)), which control the mixture component to be selected for each observation, are related through a Markov process rather than independent of each other. Recently, hidden Markov models have been generalized to pairwise Markov models and triplet Markov models which allow consideration of more complex data structures and the modelling of nonstationary data.

## Description in terms of urns

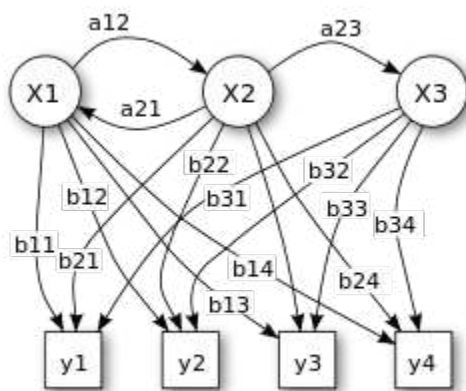


Figure 1. Probabilistic parameters of a hidden Markov model (example)

$X$  — states

$y$  — possible observations

$a$  — state transition probabilities

$b$  — output probabilities

In its discrete form, a hidden Markov process can be visualized as a generalization of the [Urn problem](#) with replacement (where each item from the urn is returned to the original urn before the next step). Consider this example: in a room that is not visible to an observer there is a genie. The room contains urns  $X_1, X_2, X_3, \dots$  each of which contains a known mix of balls, each ball labeled  $y_1, y_2, y_3, \dots$ . The genie chooses an urn in that room and randomly draws a ball from that urn. It then puts the ball onto a conveyor belt, where the observer can observe the sequence of the balls but not the sequence of urns from which they were drawn. The genie has some procedure to choose urns; the choice of the urn for the  $n$ -th ball depends only upon a random number and the choice of the urn for the  $(n - 1)$ th ball. The choice of urn does not directly depend on the urns chosen before this single previous urn; therefore, this is called a [Markov process](#). It can be described by the upper part of Figure 1.

The Markov process itself cannot be observed, only the sequence of labeled balls, thus this arrangement is called a "hidden Markov process". This is illustrated by the lower part of the diagram shown in Figure 1, where one can see that balls  $y_1, y_2, y_3, y_4$  can be drawn at each state. Even if the observer knows the composition of the urns and has just observed a sequence of three balls, e.g.  $y_1, y_2$  and  $y_3$  on the conveyor belt, the observer still cannot be *sure* which urn (*i.e.*, at which state) the genie has drawn the third ball from. However, the observer can work out other information, such as the likelihood that the third ball came from each of the urns.

## Architecture

The diagram below shows the general architecture of an instantiated HMM. Each oval shape represents a random variable that can adopt any of a number of values. The random variable  $x(t)$  is the hidden state at time  $t$  (with the model from the above diagram,  $x(t) \in \{x_1, x_2, x_3\}$ ). The random variable  $y(t)$  is the observation at time  $t$  (with  $y(t) \in \{y_1, y_2, y_3, y_4\}$ ). The arrows in the diagram (often called a [trellis diagram](#)) denote conditional dependencies.

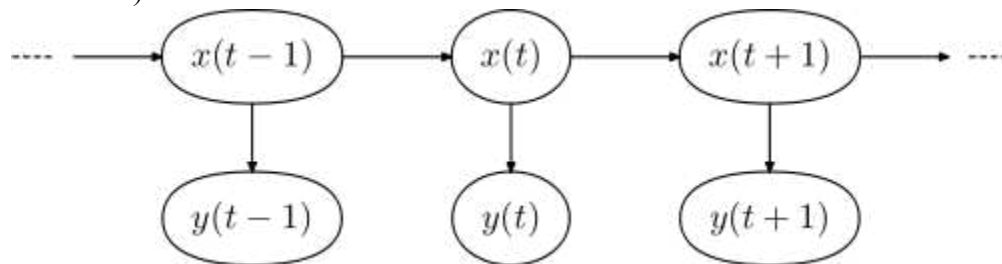
From the diagram, it is clear that the [conditional probability distribution](#) of the hidden variable  $x(t)$  at time  $t$ , given the values of the hidden variable  $x$  at all times, depends *only* on the value of the hidden variable  $x(t - 1)$ ; the values at time  $t - 2$  and before have no influence. This is called the [Markov property](#). Similarly, the value of the observed variable  $y(t)$  only depends on the value of the hidden variable  $x(t)$  (both at time  $t$ ).

In the standard type of hidden Markov model considered here, the state space of the hidden variables is discrete, while the observations themselves can either be discrete (typically generated from a [categorical distribution](#)) or continuous (typically from a [Gaussian distribution](#)). The parameters of a hidden Markov model are of two types, *transition probabilities* and *emission probabilities* (also known as *output probabilities*). The transition probabilities control the way the hidden state at time  $t$  is chosen given the hidden state at time  $t-1$ .

The hidden state space is assumed to consist of one of  $N$  possible values, modeled as a categorical distribution. (See the section below on extensions for other possibilities.) This means that for each of the  $N$  possible states that a hidden variable at time  $t$  can be in, there is a transition probability from this state to each of the  $N$  possible states of the hidden variable at time  $t-1$ , for a total of  $N^2$  transition probabilities. Note that the set of transition probabilities for transitions from any given state must sum to 1. Thus, the  $N \times N$  matrix of transition probabilities is a [Markov matrix](#). Because any one transition probability can be determined once the others are known there are a total of  $N(N-1)$  transition parameters.

In addition, for each of the  $N$  possible states, there is a set of emission probabilities governing the distribution of the observed variable at a particular time given the state of the hidden variable at

that time. The size of this set depends on the nature of the observed variable. For example, if the observed variable is discrete with  $M$  possible values, governed by a [categorical distribution](#), there will be  $M-1$  separate parameters, for a total of  $N(M-1)$  emission parameters over all hidden states. On the other hand, if the observed variable is an  $M$ -dimensional vector distributed according to an arbitrary [multivariate Gaussian distribution](#), there will be  $M$  parameters controlling the [means](#) and  $M(M+1)/2$  parameters controlling the [covariance matrix](#), for a total of  $N(M+ M(M+1)/2) = NM(M+3)/2 = O(NM^2)$  emission parameters. (In such a case, unless the value of  $M$  is small, it may be more practical to restrict the nature of the covariances between individual elements of the observation vector, e.g. by assuming that the elements are independent of each other, or less restrictively, are independent of all but a fixed number of adjacent elements.)



# Introduction to Hidden Markov Models

# Markov Models

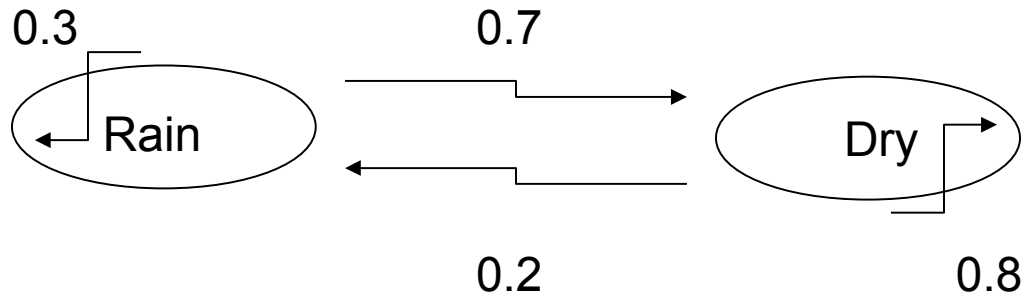
- Set of states:  $\{s_1, s_2, \dots, s_N\}$
- Process moves from one state to another generating a sequence of states :
- Markov chain property: probability of each subsequent state depends only on what was the previous state:  $s_{i1}, s_{i2}, \dots, s_{ik}, \dots$

- To define Markov model, the following probabilities have to be specified: transition probabilities  $P(s_{ik} | s_{i1}, s_{i2}, \dots, s_{ik-1})$  and initial probabilities  $P(s_{i1} | s_{i0})$

$$a_{ij} = P(s_i | s_j)$$

$$P(s_i)$$

# Example of Markov Model



- Two states : 'Rain' and 'Dry'.
- Transition probabilities:  $P(\text{'Rain'}|\text{'Rain'})=0.3$  ,  $P(\text{'Dry'}|\text{'Rain'})=0.7$  ,  
 $P(\text{'Rain'}|\text{'Dry'})=0.2$  ,  $P(\text{'Dry'}|\text{'Dry'})=0.8$
- Initial probabilities: say  $P(\text{'Rain'})=0.4$  ,  $P(\text{'Dry'})=0.6$  .

# Calculation of sequence probability

- By Markov chain property, probability of state sequence can be found by the formula:

$$P(s_{i_1}, s_{i_2}, \dots, s_{i_k}) = P(s_{i_k} | s_{i_1}, s_{i_2}, \dots, s_{i_{k-1}}) P(s_{i_1}, s_{i_2}, \dots, s_{i_{k-1}}) \\ P(s_{i_k} | s_{i_{k-1}}) P(s_{i_1}, s_{i_2}, \dots, s_{i_{k-1}}) \dots$$

- Suppose we want to calculate a probability of a sequence of states in our example, {'Dry', 'Dry', 'Rain', 'Rain'}.

$$P(\{\text{'Dry', 'Dry', 'Rain', 'Rain'}\}) = \\ P(\text{'Rain'} | \text{'Rain'}) P(\text{'Dry'} | \text{'Rain'}) P(\text{'Dry'} | \text{'Dry'}) P(\text{'Dry'}) = \\ = 0.3 * 0.2 * 0.8 * 0.6$$

# Hidden Markov models.

- Set of states:  $\{s_1, s_2, \dots, s_N\}$
- Process moves from one state to another generating a sequence of states :
- Markov chain property: probability of each subsequent state depends only on what was the previous state:  $s_{i1}, s_{i2}, \dots, s_{ik}, \dots$

- States are not visible, but each state randomly generates one of M observations (or visible states)

$$P(s_{ik} | s_{i1}, s_{i2}, \dots, s_{i\tilde{k}-1}) \quad P(s_{ik} | s_{i\tilde{k}-1})$$

- To define hidden Markov model, the following probabilities have to be specified:

matrix of transition probabilities  $A = (a_{ij}), a_{ij} = P(s_i | s_j)$ , matrix of

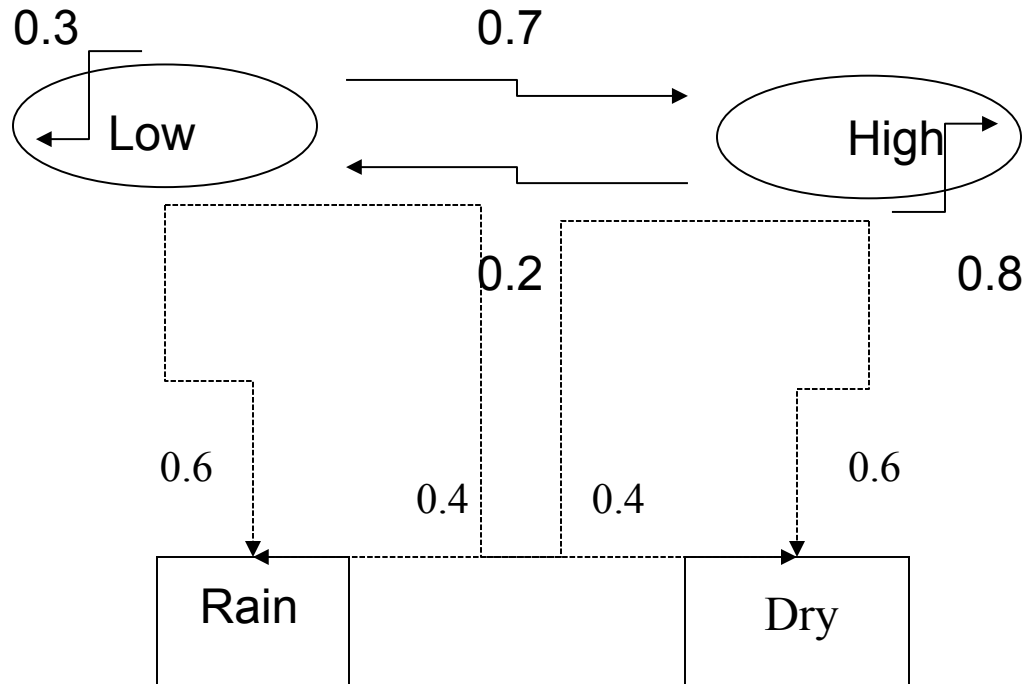
observation probabilities  $B = (b_i(v_m)), b_i(v_m) = P(v_m | s_i)$  and a

vector of initial probabilities  $\pi = (\pi_i), \pi_i = P(s_i)$ . Model is represented

by  $M = (A, B, \pi)$ .



# Example of Hidden Markov Model



# Example of Hidden Markov Model

- Two states : ‘Low’ and ‘High’ atmospheric pressure.
- Two observations : ‘Rain’ and ‘Dry’.
- Transition probabilities:  $P(\text{‘Low’}|\text{‘Low’})=0.3$  ,  $P(\text{‘High’}|\text{‘Low’})=0.7$  ,  
 $P(\text{‘Low’}|\text{‘High’})=0.2$ ,  $P(\text{‘High’}|\text{‘High’})=0.8$
- Observation probabilities :  $P(\text{‘Rain’}|\text{‘Low’})=0.6$  ,  $P(\text{‘Dry’}|\text{‘Low’})=0.4$  ,  
 $P(\text{‘Rain’}|\text{‘High’})=0.4$  ,  $P(\text{‘Dry’}|\text{‘High’})=0.3$  .
- Initial probabilities: say  $P(\text{‘Low’})=0.4$  ,  $P(\text{‘High’})=0.6$  .

# Calculation of observation sequence probability

- Suppose we want to calculate a probability of a sequence of observations in our example, {'Dry', 'Rain'}.
- Consider all possible hidden state sequences:

$$\begin{aligned} P(\{\text{'Dry'}, \text{'Rain'}\}) &= P(\{\text{'Dry'}, \text{'Rain'}\}, \{\text{'Low'}, \text{'Low'}\}) + \\ &P(\{\text{'Dry'}, \text{'Rain'}\}, \{\text{'Low'}, \text{'High'}\}) + P(\{\text{'Dry'}, \text{'Rain'}\}, \{\text{'High'}, \text{'Low'}\}) + \\ &P(\{\text{'Dry'}, \text{'Rain'}\}, \{\text{'High'}, \text{'High'}\}) \end{aligned}$$

where first term is :

$$\begin{aligned} &P(\{\text{'Dry'}, \text{'Rain'}\}, \{\text{'Low'}, \text{'Low'}\}) = \\ &P(\{\text{'Dry'}, \text{'Rain'}\} \mid \{\text{'Low'}, \text{'Low'}\}) P(\{\text{'Low'}, \text{'Low'}\}) = \\ &P(\text{'Dry'} \mid \text{'Low'}) P(\text{'Rain'} \mid \text{'Low'}) P(\text{'Low'}) P(\text{'Low'} \mid \text{'Low'}) \\ &= 0.4 * 0.4 * 0.6 * 0.4 * 0.3 \end{aligned}$$

# Main issues using HMMs :

- Evaluation problem.** Given the HMM  $M=(A, B, \pi)$  and the observation sequence  $O=O_1 O_2 \dots O_K$ , calculate the probability that model  $M$  has generated sequence  $O$ .
- **Decoding problem.** Given the HMM  $M=(A, B, \pi)$  and the observation sequence  $O=O_1 O_2 \dots O_K$ , calculate the most likely sequence of hidden states  $S_i$  that produced this observation sequence  $O$ .
- **Learning problem.** Given some training observation sequences  $O=O_1 O_2 \dots O_K$  and general structure of HMM (numbers of hidden and visible states), determine HMM parameters  $M=(A, B, \pi)$  that best fit training data.

$O=O_1 \dots O_K$  denotes a sequence of observations  $O_k \in \{v_1, \dots, v_M\}$ .



---

# Uncertain knowledge and reasoning

- Probability theory
- Bayesian networks
- Certainty factors



# 1. Probability theory

---

## 1.1 Uncertain knowledge

- .  $p \text{ symptom}(p, \text{Toothache}) \quad \text{disease}(p, \text{cavity})$
- .  $p \text{ sympt}(p, \text{Toothache})$ 
  - $\text{disease}(p, \text{cavity}) \quad \text{disease}(p, \text{gum\_disease}) \quad \dots$
- PL
  - laziness
  - theoretical ignorance
  - practical ignorance
- Probability theory  $\rightarrow$  degree of **belief** or **plausibility** of a statement – a numerical measure in  $[0,1]$
- Degree of truth – fuzzy logic    degree of belief

# 1.2 Definitions

- *Unconditional or prior probability of A* – the degree of belief in A in the absence of any other information –  $P(A)$
- A – random variable
- *Probability distribution* –  $P(A), P(A,B)$

## Example

$$P(\text{Weather} = \text{Sunny}) = 0.1$$

$$P(\text{Weather} = \text{Rain}) = 0.7$$

$$P(\text{Weather} = \text{Snow}) = 0.2$$

Weather – **random variable**

- $P(\text{Weather}) = (0.1, 0.7, 0.2)$  – probability distribution
- *Conditional probability* – posterior – once the agent has obtained some evidence B for A -  $P(A|B)$
- $P(\text{Cavity} | \text{Toothache}) = 0.8$

# Definitions - cont

- Axioms of probability
- *The measure of the occurrence of an event (random variable) A* – a function  $P:S \rightarrow R$  satisfying the axioms:
  - $0 \leq P(A) \leq 1$
  - $P(S) = 1$  ( or  $P(\text{true}) = 1$  and  $P(\text{false}) = 0$ )
  - $P(A \cup B) = P(A) + P(B) - P(A \cap B)$

$$P(A \cup \sim A) = P(A) + P(\sim A) - P(\text{false}) = P(\text{true})$$

$$P(\sim A) = 1 - P(A)$$



## Definitions - cont

A and B mutually exclusive  $\rightarrow P(A \cup B) = P(A) + P(B)$

$P(e_1 \cup e_2 \cup e_3 \cup \dots \cup e_n) = P(e_1) + P(e_2) + P(e_3) + \dots + P(e_n)$

The probability of a proposition **a** is equal to the sum of the probabilities of the atomic events in which **a** holds

$e(a)$  – the set of atomic events in which **a** holds

$$P(a) = \sum_{e_i \in e(a)} P(e_i)$$



## 1.3 Product rule

---

Conditional probabilities can be defined in terms of unconditional probabilities

*The condition probability of the occurrence of A if event B occurs*

- $P(A|B) = P(A \cap B) / P(B)$

This can be written also as:

- $P(A \cap B) = P(A|B) * P(B)$

For probability distributions

- $P(A=a1 \cap B=b1) = P(A=a1|B=b1) * P(B=b1)$

- $P(A=a1 \cap B=b2) = P(A=a1|B=b2) * P(B=b2) \dots$

- **$P(X,Y) = P(X|Y)*P(Y)$**

## 1.4 Bayes' rule and its use

---

$$P(A \cap B) = P(A|B) * P(B)$$

$$P(A \cap B) = P(B|A) * P(A)$$

Bayes' rule (theorem)

- $P(B|A) = P(A | B) * P(B) / P(A)$
- **$P(B|A) = P(A | B) * P(B) / P(A)$**



# Bayes' Theorem for 2 events

---

$$P(A_1 | B) = \frac{P(A_1)P(B | A_1)}{P(A_1)P(B | A_1) + P(A_2)P(B | A_2)}$$

$$P(A_2 | B) = \frac{P(A_2)P(B | A_2)}{P(A_1)P(B | A_1) + P(A_2)P(B | A_2)}$$



# Bayes Theorem

---

$h_i$  – hypotheses ( $i=1,k$ );

$e_1, \dots, e_n$  - evidence

$P(h_i)$

$P(h_i | e_1, \dots, e_n)$

$P(e_1, \dots, e_n | h_i)$

$$P(h_i | e_1, e_2, \dots, e_n) = \frac{P(e_1, e_2, \dots, e_n | h_i) P(h_i)}{\sum_{j=1}^k P(e_1, e_2, \dots, e_n | h_j) P(h_j)}, \quad i = 1, k$$

# Bayes' Theorem - cont



---

If  $e_1, \dots, e_n$  are independent hypotheses then

$$P(e_1, e_2, \dots, e_n | h_j) = P(e_1 | h_j) P(e_2 | h_j) \dots P(e_n | h_j), j = 1, k$$

# 1.5 Inferences

**Probability distribution**

**P(Cavity, Tooth)**

	Tooth	Tooth
Cavity	0.04	0.06
Cavity	0.01	0.89

$$P(\text{Cavity}) = 0.04 + 0.06 = 0.1$$

$$P(\text{Cavity} \quad \text{Tooth}) = 0.04 + 0.01 + 0.06 = 0.11$$

$$P(\text{Cavity} \mid \text{Tooth}) = P(\text{Cavity} \quad \text{Tooth}) / P(\text{Tooth}) = 0.04 / 0.05$$

# Inferences

	Tooth		~ Tooth	
	Catch	~ Catch	Catch	~ Catch
Cavity	0.108	0.012	0.072	0.008
~ Cavity	0.016	0.064	0.144	0.576

## Probability distributions $P(\text{Cavity, Tooth, Catch})$

$$P(\text{Cavity}) = 0.108 + 0.012 + 0.072 + 0.008 = 0.2$$

$$P(\text{Cavity} \mid \text{Tooth}) = 0.108 + 0.012 + 0.072 + 0.008 + 0.016 + 0.064 = 0.28$$

$$P(\text{Cavity} \mid \text{Tooth}) = P(\text{Cavity} \mid \text{Tooth}) / P(\text{Tooth}) = [P(\text{Cavity} \mid \text{Tooth} \mid \text{Catch}) + P(\text{Cavity} \mid \text{Tooth} \mid \sim \text{Catch})] * / P(\text{Tooth})$$





# Bayesian Networks

---

- Represent dependencies among random variables
- Give a short specification of conditional probability distribution
- Many random variables are conditionally independent
- Simplifies computations
- Graphical representation
- DAG – causal relationships among random variables
- Allows inferences based on the network structure



## 2.1 Definition of Bayesian networks

---

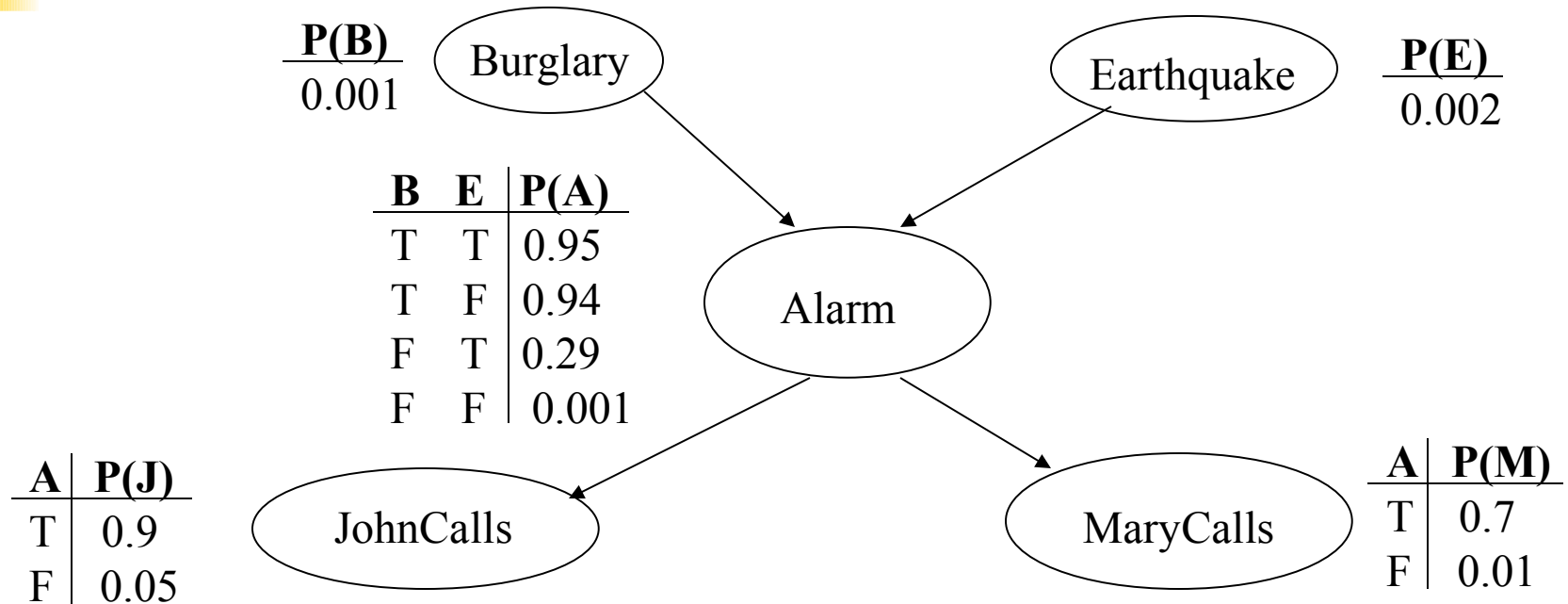
A BN is a DAG in which each node is annotated with quantitative probability information, namely:

- Nodes represent random variables (discrete or continuous)
- Directed links  $X \rightarrow Y$ :  $X$  has a direct influence on  $Y$ ,  $X$  is said to be a parent of  $Y$
- each node  $X$  has an associated conditional probability table,  $P(X_i | \text{Parents}(X_i))$  that quantify the effects of the parents on the node

Example: Weather, Cavity, Toothache, Catch

- Weather, Cavity  $\rightarrow$  Toothache, Cavity  $\rightarrow$  Catch

# Bayesian network - example



Conditional probability table

	B	E	P(A   B, E)	
			T	F
T	T	0.95	0.05	
T	F	0.94	0.06	
F	T	0.29	0.71	
F	F	0.001	0.999	



## 2.2 Bayesian network semantics

---

- A) Represent a probability distribution
- B) Specify conditional independence – build the network

A) each value of the probability distribution can be computed as:

$$P(X_1=x_1 \dots X_n=x_n) = P(x_1, \dots, x_n) = \prod_{i=1, n} P(x_i | \text{Parents}(x_i))$$

where  $\text{Parents}(x_i)$  represent the specific values of  $\text{Parents}(X_i)$

## 2.3 Building the network

$$\begin{aligned} P(X_1=x_1 \dots X_n=x_n) &= P(x_1, \dots, x_n) = \\ P(x_n | x_{n-1}, \dots, x_1) &* P(x_{n-1}, \dots, x_1) = \dots = \\ P(x_n | x_{n-1}) * P(x_{n-1} | x_{n-2}) &* \dots * P(x_2 | x_1) * P(x_1) = \\ & \prod_{i=1, n} P(x_i | x_{i-1}, \dots, x_1) \end{aligned}$$

- We can see that  $P(X_i | X_{i-1}, \dots, X_1) = P(x_i | \text{Parents}(X_i))$  if  $\text{Parents}(X_i) \subseteq \{X_{i-1}, \dots, X_1\}$
- The condition may be satisfied by labeling the nodes in an order consistent with a DAG
- Intuitively, the parents of a node  $X_i$  must be all the nodes  $X_{i-1}, \dots, X_1$  which have a direct influence on  $X_i$ .

# Building the network - cont

- Pick a set of random variables that describe the problem
- Pick an ordering of those variables
- **while** there are still variables **repeat**
  - (a) choose a variable  $X_i$  and add a node associated to  $X_i$
  - (b) assign  $\text{Parents}(X_i)$  a minimal set of nodes that already exists in the network such that the conditional independence property is satisfied
  - (c) define the conditional probability table for  $X_i$
- Because each node is linked only to previous nodes DAG
- $P(\text{MaryCalls} \mid \text{JohnCalls}, \text{Alarm}, \text{Burglary}, \text{Earthquake}) = P(\text{MaryCalls} \mid \text{Alarm})$

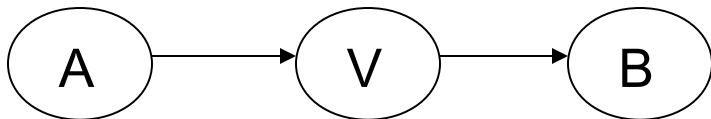


# Compactness of node ordering

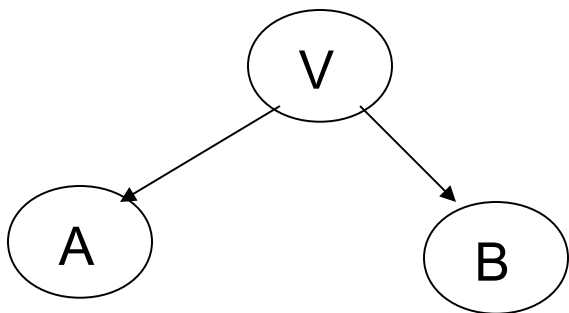
---

- Far more compact than a probability distribution
- Example of **locally structured system** (or *sparse*): each component interacts directly only with a limited number of other components
- Associated usually with a linear growth in complexity rather than with an exponential one
- *The order of adding the nodes is important*
- The correct order in which to add nodes is to add the “root causes” first, then the variables they influence, and so on, until we reach the leaves

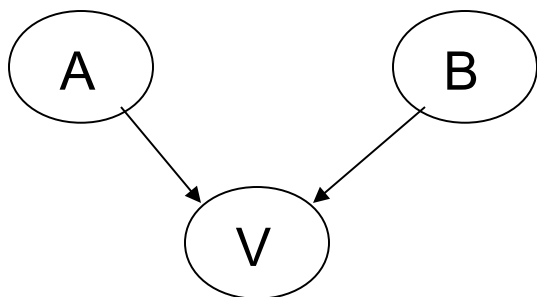
## 2.4 Probabilistic inferences



$$P(A \quad V \quad B) = P(A) * P(V|A) * P(B|V)$$



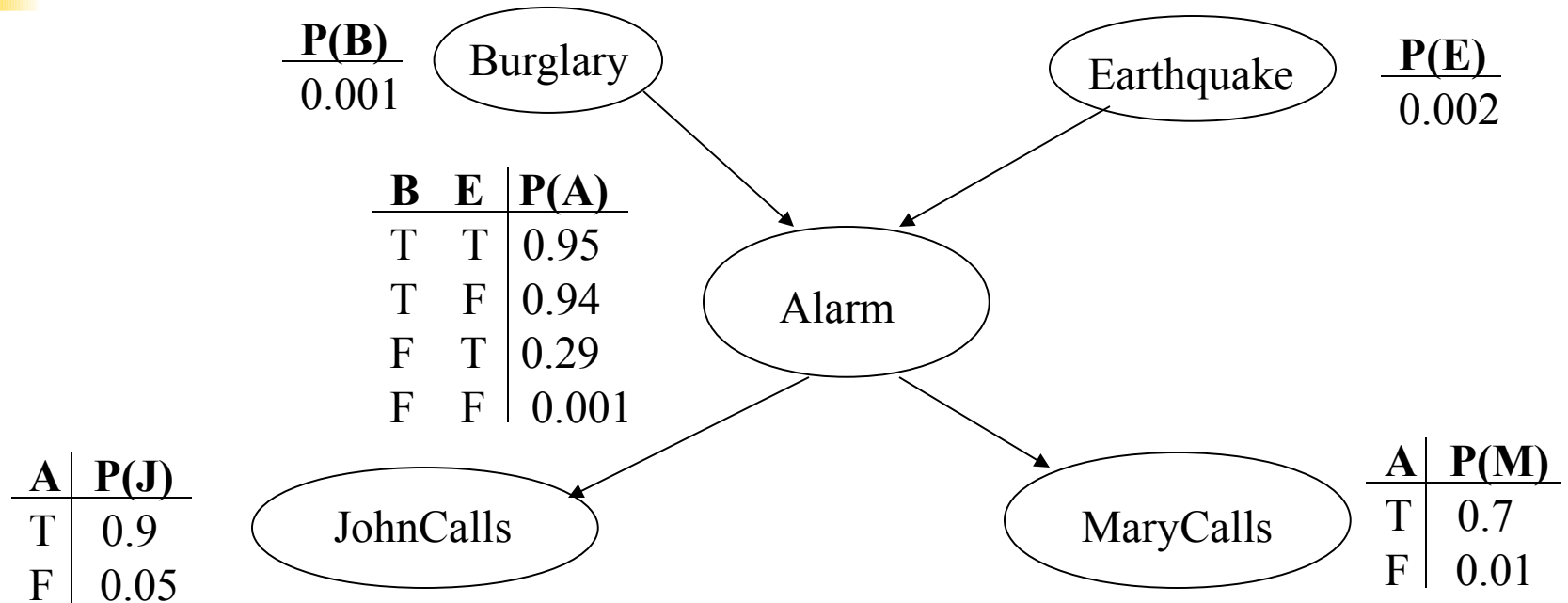
$$P(A \quad V \quad B) = P(V) * P(A|V) * P(B|V)$$



$$P(A \quad V \quad B) = P(A) * P(B) * P(V|A,B)$$

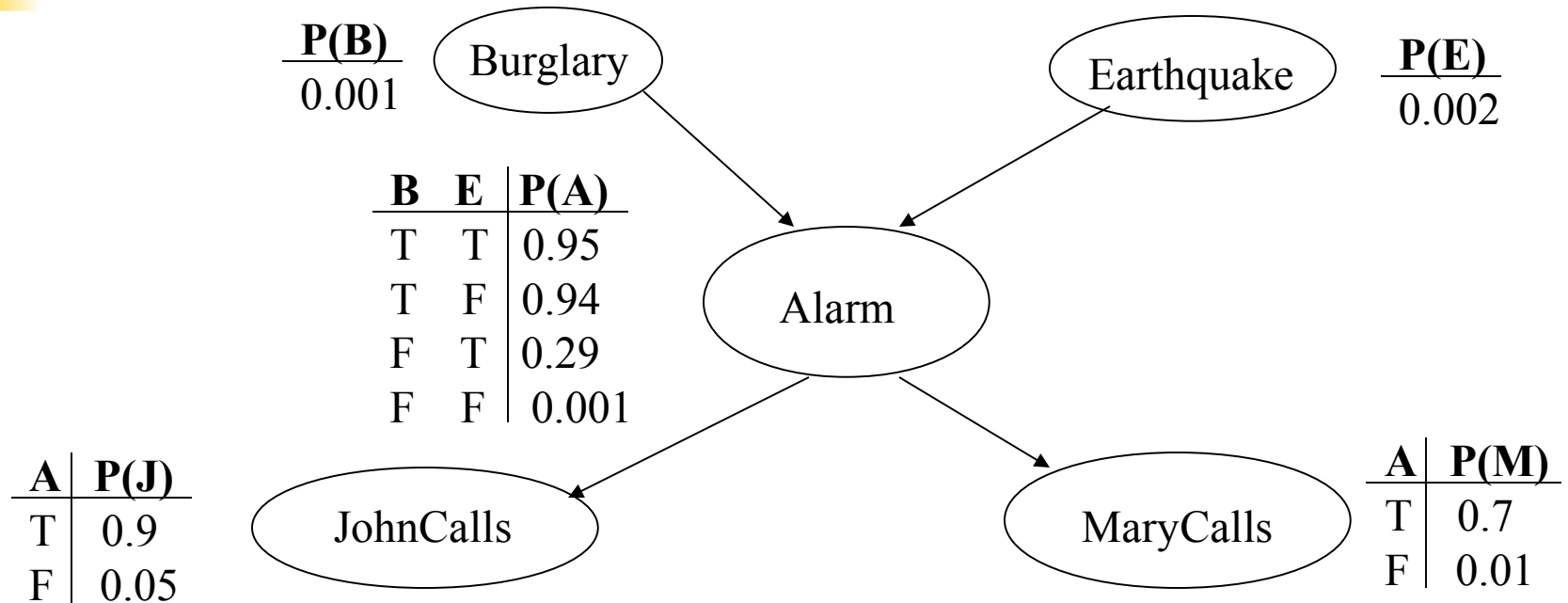


# Probabilistic inferences



$$\begin{aligned}
 &P(J \ M \ A \ B \ E) = \\
 &P(J|A) * P(M|A) * P(A| \ B \ E) * P( \ B) \ P( \ E) = \\
 &0.9 * 0.7 * 0.001 * 0.999 * 0.998 = 0.00062
 \end{aligned}$$

# Probabilistic inferences



$$\begin{aligned}
 P(A|B) &= P(A|B,E) * P(E|B) + P(A|B, \bar{E}) * P(\bar{E}|B) \\
 &= P(A|B,E) * P(E) + P(A|B, \bar{E}) * P(\bar{E}) \\
 &= 0.95 * 0.002 + 0.94 * 0.998 = 0.94002
 \end{aligned}$$

## 2.5 Different types of inferences

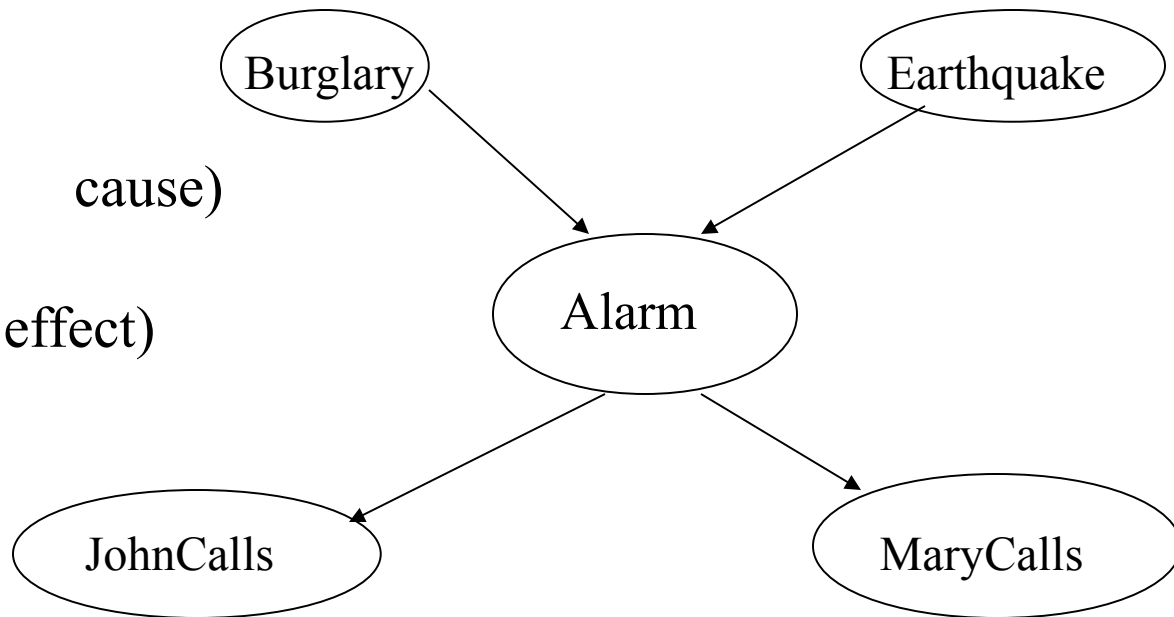
**Diagnosis inferences** (effect cause)

$P(\text{Burglary} \mid \text{JohnCalls})$

**Causal inferences** (cause effect)

$P(\text{JohnCalls} \mid \text{Burglary}),$

$P(\text{MaryCalls} \mid \text{Burglary})$



**Intercausal inferences** (between cause and common effects)

$P(\text{Burglary} \mid \text{Alarm} \quad \text{Earthquake})$

**Mixed inferences**

$P(\text{Alarm} \mid \text{JohnCalls} \quad \text{Earthquake})$  diag + causal

$P(\text{Burglary} \mid \text{JohnCalls} \quad \text{Earthquake})$  diag + intercausal



# 3. Certainty factors

---

- The MYCIN model
- Certainty factors / Confidence coefficients (CF)
- Heuristic model of uncertain knowledge
- In MYCIN – two probabilistic functions to model the degree of belief and the degree of disbelief in a hypothesis
  - *function to measure the degree of belief* - MB
  - *function to measure the degree of disbelief* - MD
- **MB[h,e]** – how much the belief in **h** increases based on evidence **e**
- **MD[h,e]** - how much the disbelief in **h** increases based on evidence **e**

# 3.1 Belief functions

$$MB[h, e] = \frac{\max(P(h | e), P(h)) \tilde{P}(h)}{\max(0, 1) \tilde{P}(h)}$$

daca  $P(h) = 1$   
in caz contrar

$$MD[h, e] = \frac{\min(P(h | e), P(h)) \tilde{P}(h)}{\min(0, 1) \tilde{P}(h)}$$

daca  $P(h) = 0$   
in caz contrar

- *Certainty factor*

$$CF[h, e] = MB[h, e] \tilde{P}(h) - MD[h, e]$$

# Belief functions - features

- Value range

$$0 \leq MB[h,e] \leq 1 \quad 0 \leq MD[h,e] \leq 1 \quad \tilde{1} \leq CF[h,e] \leq 1$$

- If  $h$  is sure, i.e.  $P(h|e) = 1$ , then

$$MB[h,e] = \frac{1 \cdot P(h)}{1 \cdot P(h)} = 1 \quad MD[h,e] = 0 \quad CF[h,e] = 1$$

- If the negation of  $h$  is sure, i.e.  $P(h|e) = 0$  then

$$MB[h,e] = 0 \quad MD[h,e] = \frac{0 \cdot P(h)}{0 \cdot P(h)} = 1 \quad CF[h,e] = \tilde{1}$$



# Example in MYCIN

---

- **if** (1) the type of the organism is gram-positive, and
- (2) the morphology of the organism is coccus, and
- (3) the growth of the organism is chain
- **then** there is a strong evidence (0.7) that the identity of the organism is streptococcus

Example of facts in MYCIN :

- (identity organism-1 pseudomonas 0.8)
- (identity organism-2 e.coli 0.15)
- (morphology organism-2 coccus 1.0)



## 3.2 Combining belief functions

---

### (1) Incremental gathering of evidence

- The same attribute value,  $h$ , is obtained by two separate paths of inference, with two separate CFs :  $CF[h,s1]$  si  $CF[h,s2]$
- The two different paths, corresponding to hypotheses  $s1$  and  $s2$  may be different braches of the search tree.
- $CF[h, s1\&s2] = CF[h,s1] + CF[h,s2] - CF[h,s1]*CF[h,s2]$
- (identity organism-1 pseudomonas 0.8)
- (identity organism-1 pseudomonas 0.7)





# Combining belief functions

---

## (2) Conjunction of hypothesis

- Applied for computing the CF associated to the premises of a rule which has several conditions

**if**  $A = a1$  and  $B = b1$  **then** ...

WM:  $(A\ a1\ h1\ cf1)(B\ b1\ h2\ cf2)$

- $CF[h1\&h2, s] = \min(CF[h1,s], CF[h2,s])$



# Combining belief functions

---

## (3) Combining beliefs

- An uncertain value is deduced based on a rule which has as input conditions based on uncertain values (may be obtained by applying other rules for example).
- Allows the computation of the CF of the fact deduced by the rule based on the rule's CF and the CF of the hypotheses
- $CF[s,e]$  – belief in a hypothesis  $s$  based on previous evidence  $e$
- $CF[h,s]$  - CF in  $h$  if  $s$  is sure
- $CF'[h,s] = CF[h,s] * CF [s,e]$



# Combining belief functions

---

## (3) Combining beliefs – cont

**if**  $A = a1$  and  $B = b1$  **then**  $C = c1$  0.7

ML: ( $A$   $a1$  0.9)    ( $B$   $b1$  0.6)

$CF(\text{premises}) = \min(0.9, 0.6) = 0.6$

**$CF(\text{conclusion}) = CF(\text{premises}) * CF(\text{rule}) = 0.6 * 0.7$**

**ML: ( $C$   $c1$  0.42)**



## 3.3 Limits of CF

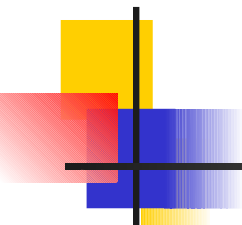
---

- CF of MYCIN assumes that that the hypothesis are sustained by independent evidence
- An example shows what happens if this condition is violated

A: The sprinkle functioned last night

U: The grass is wet in the morning

P: Last night it rained



---

R1: **if** the sprinkle functioned last night  
**then** there is a strong evidence (0.9) that the grass is wet in the morning

R2: **if** the grass is wet in the morning  
**then** there is a strong evidence (0.8) that it rained last night

- $CF[U,A] = 0.9$
- therefore the evidence sprinkle sustains the hypothesis wet grass with  $CF = 0.9$
  
- $CF[P,U] = 0.8$
- therefore the evidence wet grass sustains the hypothesis rain with  $CF = 0.8$
  
- $CF[P,A] = 0.8 * 0.9 = 0.72$
- therefore the evidence sprinkle sustains the hypothesis rain with  $CF = 0.72$

Solutions

# 2

---

## Introducing Bayesian Networks

---

### 2.1 Introduction

Having presented both theoretical and practical reasons for artificial intelligence to use probabilistic reasoning, we now introduce the key computer technology for dealing with probabilities in AI, namely **Bayesian networks**. Bayesian networks (BNs) are graphical models for reasoning under uncertainty, where the nodes represent variables (discrete or continuous) and arcs represent direct connections between them. These direct connections are often causal connections. In addition, BNs model the quantitative strength of the connections between variables, allowing probabilistic beliefs about them to be updated automatically as new information becomes available.

In this chapter we will describe how Bayesian networks are put together (the **syntax**) and how to interpret the information encoded in a network (the **semantics**). We will look at how to model a problem with a Bayesian network and the types of reasoning that can be performed.

---

### 2.2 Bayesian network basics

A **Bayesian network** is a graphical structure that allows us to represent and reason about an uncertain domain. The nodes in a Bayesian network represent a set of random variables,  $\mathbf{X} = X_1, \dots, X_i, \dots, X_n$ , from the domain. A set of directed **arcs** (or links) connects pairs of nodes,  $X_i \rightarrow X_j$ , representing the direct dependencies between variables. Assuming discrete variables, the strength of the relationship between variables is quantified by conditional probability distributions associated with each node. The only constraint on the arcs allowed in a BN is that there must not be any directed cycles: you cannot return to a node simply by following directed arcs. Such networks are called directed acyclic graphs, or simply **dags**.

There are a number of steps that a **knowledge engineer**<sup>1</sup> must undertake when building a Bayesian network. At this stage we will present these steps as a sequence; however it is important to note that in the real-world the process is not so simple. In Chapter 10 we provide a fuller description of BN knowledge engineering.

---

<sup>1</sup>Knowledge engineer in the jargon of AI means a practitioner applying AI technology.

Throughout the remainder of this section we will use the following simple medical diagnosis problem.

**Example problem: Lung cancer.** *A patient has been suffering from shortness of breath (called dyspnoea) and visits the doctor, worried that he has lung cancer. The doctor knows that other diseases, such as tuberculosis and bronchitis, are possible causes, as well as lung cancer. She also knows that other relevant information includes whether or not the patient is a smoker (increasing the chances of cancer and bronchitis) and what sort of air pollution he has been exposed to. A positive X-ray would indicate either TB or lung cancer.*<sup>2</sup>

### 2.2.1 Nodes and values

First, the knowledge engineer must identify the variables of interest. This involves answering the question: what are the nodes to represent and what values can they take, or what state can they be in? For now we will consider only nodes that take discrete values. The values should be both **mutually exclusive** and **exhaustive**, which means that the variable must take on exactly one of these values at a time. Common types of discrete nodes include:

- Boolean nodes, which represent propositions, taking the binary values true ( $T$ ) and false ( $F$ ). In a medical diagnosis domain, the node *Cancer* would represent the proposition that a patient has cancer.
- Ordered values. For example, a node *Pollution* might represent a patient's pollution exposure and take the values  $\{low, medium, high\}$ .
- Integral values. For example, a node called *Age* might represent a patient's age and have possible values from 1 to 120.

Even at this early stage, modeling choices are being made. For example, an alternative to representing a patient's exact age might be to clump patients into different age groups, such as  $\{baby, child, adolescent, young, middleaged, old\}$ . The trick is to choose values that represent the domain efficiently, but with enough detail to perform the reasoning required. More on this later!

**TABLE 2.1**  
Preliminary choices of nodes and values for the lung cancer example.

Node name	Type	Values
<i>Pollution</i>	Binary	$\{low, high\}$
<i>Smoker</i>	Boolean	$\{T, F\}$
<i>Cancer</i>	Boolean	$\{T, F\}$
<i>Dyspnoea</i>	Boolean	$\{T, F\}$
<i>X-ray</i>	Binary	$\{pos, neg\}$

<sup>2</sup>This is a modified version of the so-called "Asia" problem Lauritzen and Spiegelhalter, 1988, given in §2.5.3.

For our example, we will begin with the restricted set of nodes and values shown in Table 2.1. These choices already limit what can be represented in the network. For instance, there is no representation of other diseases, such as TB or bronchitis, so the system will not be able to provide the probability of the patient having them. Another limitation is a lack of differentiation, for example between a heavy or a light smoker, and again the model assumes at least some exposure to pollution. Note that all these nodes have only two values, which keeps the model simple, but in general there is no limit to the number of discrete values.

### 2.2.2 Structure

The structure, or topology, of the network should capture qualitative relationships between variables. In particular, two nodes should be connected directly if one affects or causes the other, with the arc indicating the direction of the effect. So, in our medical diagnosis example, we might ask what factors affect a patient’s chance of having cancer? If the answer is “Pollution and smoking,” then we should add arcs from *Pollution* and *Smoker* to *Cancer*. Similarly, having cancer will affect the patient’s breathing and the chances of having a positive X-ray result. So we add arcs from *Cancer* to *Dyspnoea* and *XRay*. The resultant structure is shown in Figure 2.1. It is important to note that this is just one possible structure for the problem; we look at alternative network structures in §2.4.3.

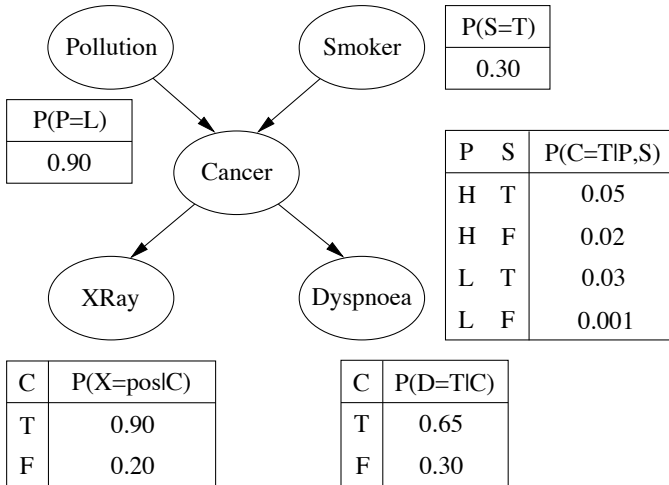


FIGURE 2.1: A BN for the lung cancer problem.

#### Structure terminology and layout

In talking about network structure it is useful to employ a family metaphor: a node is a **parent** of a **child**, if there is an arc from the former to the latter. Extending the



metaphor, if there is a directed chain of nodes, one node is an **ancestor** of another if it appears earlier in the chain, whereas a node is a **descendant** of another node if it comes later in the chain. In our example, the *Cancer* node has two parents, *Pollution* and *Smoker*, while *Smoker* is an ancestor of both *X-ray* and *Dyspnoea*. Similarly, *X-ray* is a child of *Cancer* and descendant of *Smoker* and *Pollution*. The set of parent nodes of a node  $X$  is given by  $Parents(X)$ .

Another useful concept is that of the **Markov blanket** of a node, which consists of the node's parents, its children, and its children's parents. Other terminology commonly used comes from the "tree" analogy (even though Bayesian networks in general are graphs rather than trees): any node without parents is called a **root** node, while any node without children is called a **leaf** node. Any other node (non-leaf and non-root) is called an **intermediate node**. Given a causal understanding of the BN structure, this means that root nodes represent original causes, while leaf nodes represent final effects. In our cancer example, the causes *Pollution* and *Smoker* are root nodes, while the effects *X-ray* and *Dyspnoea* are leaf nodes.

By convention, for easier visual examination of BN structure, networks are usually laid out so that the arcs generally point from top to bottom. This means that the BN "tree" is usually depicted upside down, with roots at the top and leaves at the bottom!<sup>3</sup>

### 2.2.3 Conditional probabilities

Once the topology of the BN is specified, the next step is to quantify the relationships between connected nodes – this is done by specifying a conditional probability distribution for each node. As we are only considering discrete variables at this stage, this takes the form of a conditional probability *table* (CPT).

First, for each node we need to look at all the possible combinations of values of those parent nodes. Each such combination is called an **instantiation** of the parent set. For each distinct instantiation of parent node values, we need to specify the probability that the child will take each of its values.

For example, consider the *Cancer* node of Figure 2.1. Its parents are *Pollution* and *Smoking* and take the possible joint values  $\{ \langle H, T \rangle, \langle H, F \rangle, \langle L, T \rangle, \langle L, F \rangle \}$ . The conditional probability table specifies in order the probability of cancer for each of these cases to be:  $\langle 0.05, 0.02, 0.03, 0.001 \rangle$ . Since these *are* probabilities, and must sum to one over all possible states of the *Cancer* variable, the probability of no cancer is already implicitly given as one minus the above probabilities in each case; i.e., the probability of no cancer in the four possible parent instantiations is  $\langle 0.95, 0.98, 0.97, 0.999 \rangle$ .

Root nodes also have an associated CPT, although it is degenerate, containing only one row representing its prior probabilities. In our example, the prior for a patient being a smoker is given as 0.3, indicating that 30% of the population that the

---

<sup>3</sup>Oddly, this is the antipodean standard in computer science; we'll let you decide what that may mean about computer scientists!

doctor sees are smokers, while 90% of the population are exposed to only low levels of pollution.

Clearly, if a node has many parents or if the parents can take a large number of values, the CPT can get very large! The size of the CPT is, in fact, exponential in the number of parents. Thus, for Boolean networks a variable with  $n$  parents requires a CPT with  $2^{n+1}$  probabilities.

### 2.2.4 The Markov property

In general, modeling with Bayesian networks requires the assumption of the **Markov property**: there are no direct dependencies in the system being modeled which are not already explicitly shown via arcs. In our *Cancer* case, for example, there is no way for smoking to influence dyspnoea except by way of causing cancer (or not) — there is no hidden “backdoor” from smoking to dyspnoea. Bayesian networks which have the Markov property are also called **Independence-maps** (or, **I-maps** for short), since every independence suggested by the lack of an arc is real in the system.

Whereas the independencies suggested by a lack of arcs are generally required to exist in the system being modeled, it is not generally required that the arcs in a BN correspond to real dependencies in the system. The CPTs may be parameterized in such a way as to nullify any dependence. Thus, for example, every fully-connected Bayesian network can represent, perhaps in a wasteful fashion, any joint probability distribution over the variables being modeled. Of course, we shall prefer **minimal models** and, in particular, **minimal I-maps**, which are I-maps such that the deletion of any arc violates I-mapness by implying a non-existent independence in the system.

If, in fact, every arc in a BN happens to correspond to a direct dependence in the system, then the BN is said to be a **Dependence-map** (or, **D-map** for short). A BN which is both an I-map and a D-map is said to be a **perfect map**.

---

## 2.3 Reasoning with Bayesian networks

Now that we know how a domain and its uncertainty may be represented in a Bayesian network, we will look at how to use the Bayesian network to reason about the domain. In particular, when we observe the value of some variable, we would like to **condition** upon the new information. The process of conditioning (also called **probability propagation** or **inference** or **belief updating**) is performed via a “flow of information” through the network. Note that this information flow is *not* limited to the directions of the arcs. In our probabilistic system, this becomes the task of computing the posterior probability distribution for a set of **query** nodes, given values for some **evidence** (or **observation**) nodes.

### 2.3.1 Types of reasoning

Bayesian networks provide full representations of probability distributions over their variables. That implies that they can be conditioned upon any subset of their variables, supporting any direction of reasoning.

For example, one can perform **diagnostic reasoning**, i.e., reasoning from symptoms to cause, such as when a doctor observes *Dyspnoea* and then updates his belief about *Cancer* and whether the patient is a *Smoker*. Note that this reasoning occurs in the *opposite* direction to the network arcs.

Or again, one can perform **predictive reasoning**, reasoning from new information about causes to new beliefs about effects, following the directions of the network arcs. For example, the patient may tell his physician that he is a smoker; even before any symptoms have been assessed, the physician knows this will increase the chances of the patient having cancer. It will also change the physician's expectations that the patient will exhibit other symptoms, such as shortness of breath or having a positive X-ray result.

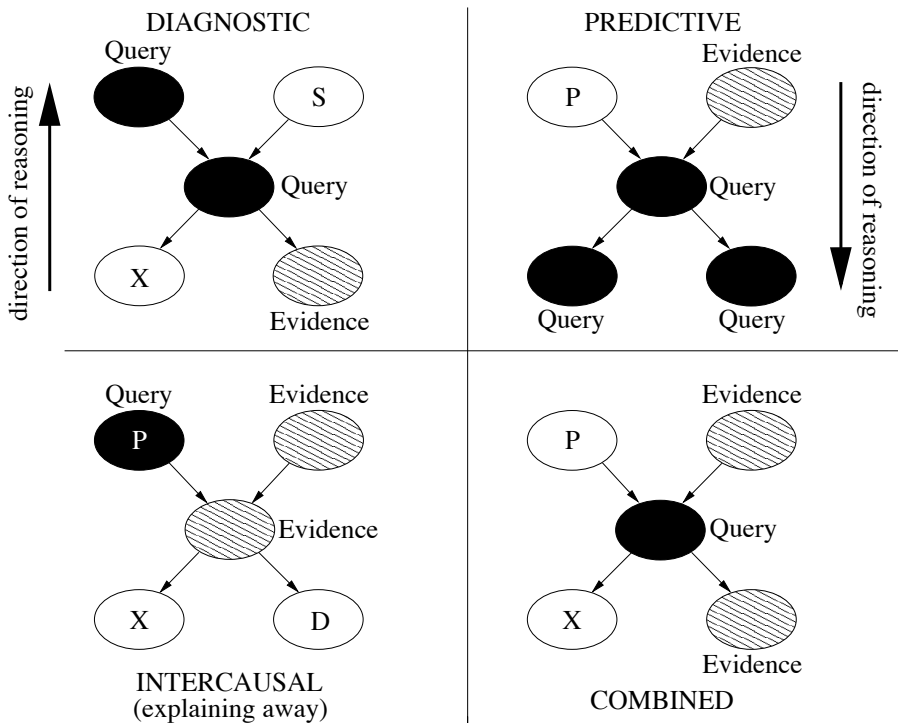


FIGURE 2.2: Types of reasoning.

A further form of reasoning involves reasoning about the mutual causes of a common effect; this has been called **intercausal reasoning**. A particular type called

**explaining away** is of some interest. Suppose that there are exactly two possible causes of a particular effect, represented by a **v-structure** in the BN. This situation occurs in our model of Figure 2.1 with the causes *Smoker* and *Pollution* which have a common effect, *Cancer* (of course, reality is more complex than our example!). Initially, according to the model, these two causes are independent of each other; that is, a patient smoking (or not) does not change the probability of the patient being subject to pollution. Suppose, however, that we learn that Mr. Smith has cancer. This will raise our probability for both possible causes of cancer, increasing the chances both that he is a smoker and that he has been exposed to pollution. Suppose then that we discover that he is a smoker. This new information explains the observed cancer, which in turn *lowers* the probability that he has been exposed to high levels of pollution. So, even though the two causes are initially independent, with knowledge of the effect the presence of one explanatory cause renders an alternative cause less likely. In other words, the alternative cause has been *explained away*.

Since any nodes may be query nodes and any may be evidence nodes, sometimes the reasoning does not fit neatly into one of the types described above. Indeed, we can combine the above types of reasoning in any way. Figure 2.2 shows the different varieties of reasoning using the Cancer BN. Note that the last combination shows the simultaneous use of diagnostic and predictive reasoning.

### 2.3.2 Types of evidence

So Bayesian networks can be used for calculating new beliefs when new information – which we have been calling **evidence** – is available. In our examples to date, we have considered evidence as a definite finding that a node  $X$  has a particular value,  $x$ , which we write as  $X = x$ . This is sometimes referred to as **specific evidence**. For example, suppose we discover the patient is a smoker, then  $Smoker = T$ , which is specific evidence.

However, sometimes evidence is available that is not so definite. The evidence might be that a node  $Y$  has the value  $y_1$  or  $y_2$  (implying that all other values are impossible). Or the evidence might be that  $Y$  is *not* in state  $y_1$  (but may take any of its other values); this is sometimes called a **negative evidence**.

In fact, the new information might simply be any new probability distribution over  $Y$ . Suppose, for example, that the radiologist who has taken and analyzed the X-ray in our cancer example is uncertain. He thinks that the X-ray looks positive, but is only 80% sure. Such information can be incorporated equivalently to Jeffrey conditionalization of §1.5.1, in which case it would correspond to adopting a new posterior distribution for the node in question. In Bayesian networks this is also known as **virtual evidence**. Since it is handled via likelihood information, it is also known as **likelihood evidence**. We defer further discussion of virtual evidence until Chapter 3, where we can explain it through the effect on belief updating.

### 2.3.3 Reasoning with numbers

Now that we have described qualitatively the types of reasoning that are possible using BNs, and types of evidence, let's look at the actual numbers. Even before we obtain any evidence, we can compute a prior belief for the value of each node; this is the node's prior probability distribution. We will use the notation  $\text{Bel}(X)$  for the posterior probability distribution over a variable  $X$ , to distinguish it from the prior and conditional probability distributions (i.e.,  $P(X)$ ,  $P(X|Y)$ ).

The exact numbers for the updated beliefs for each of the reasoning cases described above are given in Table 2.2. The first set are for the priors and conditional probabilities originally specified in Figure 2.1. The second set of numbers shows what happens if the smoking rate in the population increases from 30% to 50%, as represented by a change in the prior for the *Smoker* node. Note that, since the two cases differ only in the prior probability of smoking ( $P(S = T) = 0.3$  versus  $P(S = T) = 0.5$ ), when the evidence itself is about the patient being a smoker, then the prior becomes irrelevant and both networks give the same numbers.

**TABLE 2.2**

Updated beliefs given new information with smoking rate 0.3 (top set) and 0.5 (bottom set).

Node P(S)=0.3	No Evidence	Reasoning Case				
		Diagnostic D=T	Predictive S=T	Intercausal		Combined
				C=T	C=T S=T	D=T S=T
Bel(P=high)	0.100	0.102	0.100	0.249	0.156	0.102
Bel(S=T)	0.300	0.307	1	0.825	1	1
Bel(C=T)	0.011	0.025	0.032	1	1	0.067
Bel(X=pos)	0.208	0.217	0.222	0.900	0.900	0.247
Bel(D=T)	0.304	1	0.311	0.650	0.650	1
P(S)=0.5						
Bel(P=high)	0.100	0.102	0.100	0.201	0.156	0.102
Bel(S=T)	0.500	0.508	1	0.917	1	1
Bel(C=T)	0.174	0.037	0.032	1	1	0.067
Bel(X=pos)	0.212	0.226	0.311	0.900	0.900	0.247
Bel(D=T)	0.306	1	0.222	0.650	0.650	1

Belief updating can be done using a number of exact and approximate inference algorithms. We give details of these algorithms in Chapter 3, with particular emphasis on how choosing different algorithms can affect the efficiency of both the knowledge engineering process and the automated reasoning in the deployed system. However, most existing BN software packages use essentially the same algorithm and it is quite possible to build and use BNs without knowing the details of the belief updating algorithms.

## 2.4 Understanding Bayesian networks

We now consider how to interpret the information encoded in a BN — the probabilistic **semantics** of Bayesian networks.

### 2.4.1 Representing the joint probability distribution

Most commonly, BNs are considered to be representations of joint probability distributions. There is a fundamental assumption that there is a useful underlying structure to the problem being modeled that can be captured with a BN, i.e., that not every node is connected to every other node. If such domain structure exists, a BN gives a more compact representation than simply describing the probability of every joint instantiation of all variables. **Sparse** Bayesian networks (those with relatively few arcs, which means few parents for each node) represent probability distributions in a computationally tractable way.

Consider a BN containing the  $n$  nodes,  $X_1$  to  $X_n$ , taken in that order. A particular value in the joint distribution is represented by  $P(X_1 = x_1, X_2 = x_2, \dots, X_n = x_n)$ , or more compactly,  $P(x_1, x_2, \dots, x_n)$ . The **chain rule** of probability theory allows us to factorize joint probabilities so:

$$\begin{aligned} P(x_1, x_2, \dots, x_n) &= P(x_1) \times P(x_2|x_1) \dots \times P(x_n|x_1, \dots, x_{n-1}) \\ &= \prod_i P(x_i|x_1, \dots, x_{i-1}) \end{aligned} \quad (2.1)$$

Recalling from §2.2.4 that the structure of a BN implies that the value of a particular node is conditional *only* on the values of its parent nodes, this reduces to

$$P(x_1, x_2, \dots, x_n) = \prod_i P(x_i | \text{Parents}(X_i))$$

provided  $\text{Parents}(X_i) \subseteq \{X_1, \dots, X_{i-1}\}$ . For example, by examining Figure 2.1, we can simplify its joint probability expressions. E.g.,

$$\begin{aligned} P(X = \text{pos} \wedge D = T \wedge C = T \wedge P = \text{low} \wedge S = F) \\ &= P(X = \text{pos} | D = T, C = T, P = \text{low}, S = F) \\ &\quad \times P(D = T | C = T, P = \text{low}, S = F) \\ &\quad \times P(C = T | P = \text{low}, S = F) P(P = \text{low} | S = F) P(S = F) \\ &= P(X = \text{pos} | C = T) P(D = T | C = T) P(C = T | P = \text{low}, S = F) \\ &\quad \times P(P = \text{low}) P(S = F) \end{aligned}$$

### 2.4.2 Pearl's network construction algorithm

The condition that  $\text{Parents}(X_i) \subseteq \{X_1, \dots, X_{i-1}\}$  allows us to construct a network from a given ordering of nodes using Pearl's network construction algorithm (1988,

section 3.3). Furthermore, the resultant network will be a unique minimal I-map, assuming the probability distribution is positive. The construction algorithm (Algorithm 2.1) simply processes each node in order, adding it to the existing network and adding arcs from a minimal set of parents such that the parent set renders the current node conditionally independent of every other node preceding it.

**Algorithm 2.1** Pearl’s Network Construction Algorithm

1. Choose the set of relevant variables  $\{X_i\}$  that describe the domain.
2. Choose an ordering for the variables,  $\langle X_1, \dots, X_n \rangle$ .
3. While there are variables left:
  - (a) Add the next variable  $X_i$  to the network.
  - (b) Add arcs to the  $X_i$  node from some minimal set of nodes already in the net,  $\text{Parents}(X_i)$ , such that the following conditional independence property is satisfied:

$$P(X_i|X'_1, \dots, X'_m) = P(X_i|\text{Parents}(X_i))$$

where  $X'_1, \dots, X'_m$  are all the variables preceding  $X_i$ .

- (c) Define the CPT for  $X_i$ .

### 2.4.3 Compactness and node ordering

Using this construction algorithm, it is clear that a different node order may result in a different network structure, with both nevertheless representing the same joint probability distribution.

In our example, several different orderings will give the original network structure: *Pollution* and *Smoker* must be added first, but in either order, then *Cancer*, and then *Dyspnoea* and *X-ray*, again in either order.

On the other hand, if we add the symptoms first, we will get a markedly different network. Consider the order  $\langle D, X, C, P, S \rangle$ .  $D$  is now the new root node. When adding  $X$ , we must consider “Is *X-ray* independent of *Dyspnoea*?” Since they have a common cause in *Cancer*, they will be dependent: learning the presence of one symptom, for example, raises the probability of the other being present. Hence, we have to add an arc from  $D$  to  $X$ . When adding *Cancer*, we note that *Cancer* is directly dependent upon both *Dyspnoea* and *X-ray*, so we must add arcs from both. For *Pollution*, an arc is required from  $C$  to  $P$  to carry the direct dependency. When the final node, *Smoker*, is added, not only is an arc required from  $C$  to  $S$ , but another from  $P$  to  $S$ . In our story  $S$  and  $P$  are independent, but in the new network, without this final arc,  $P$  and  $S$  are made dependent by having a common cause, so that effect must be counterbalanced by an additional arc. The result is two additional arcs and three new probability values associated with them, as shown in Figure 2.3(a). Given the order  $\langle D, X, P, S, C \rangle$ , we get Figure 2.3(b), which is fully connected and requires as many CPT entries as a brute force specification of the full joint distribution! In such cases, the use of Bayesian networks offers no representational, or computational, advantage.

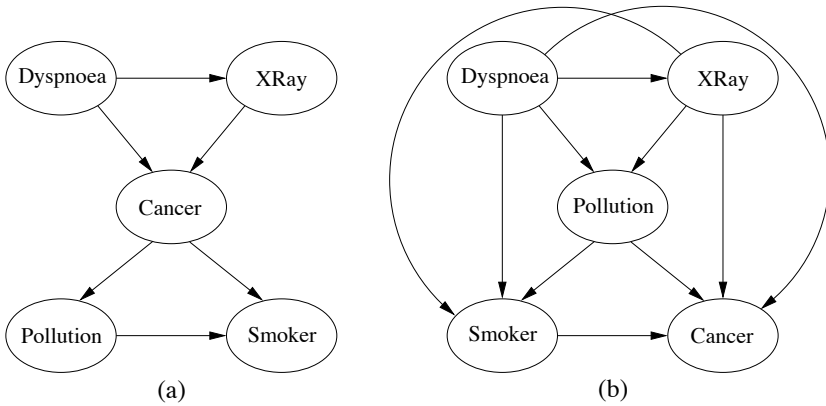


FIGURE 2.3: Alternative structures obtained using Pearl’s network construction algorithm with orderings: (a)  $\langle D, X, C, P, S \rangle$ ; (b)  $\langle D, X, P, S, C \rangle$ .

It is desirable to build the most compact BN possible, for three reasons. First, the more compact the model, the more tractable it is. It will have fewer probability values requiring specification; it will occupy less computer memory; probability updates will be more computationally efficient. Second, overly dense networks fail to represent independencies explicitly. And third, overly dense networks fail to represent the *causal* dependencies in the domain. We will discuss these last two points just below.

We can see from the examples that the compactness of the BN depends on getting the node ordering “right.” The optimal order is to add the root causes first, then the variable(s) they influence directly, and continue until leaves are reached.<sup>4</sup> To understand *why*, we need to consider the relation between probabilistic and causal dependence.

## 2.4.4 Conditional independence

Bayesian networks which satisfy the Markov property (and so are I-maps) explicitly express conditional independencies in probability distributions. The relation between conditional independence and Bayesian network structure is important for understanding how BNs work.

### 2.4.4.1 Causal chains

Consider a causal chain of three nodes, where  $A$  causes  $B$  which in turn causes  $C$ , as shown in Figure 2.4(a). In our medical diagnosis example, one such causal chain is “smoking causes cancer which causes dyspnoea.” Causal chains give rise to condi-

<sup>4</sup>Of course, one may not know the causal order of variables. In that case the automated discovery methods discussed in Part II may be helpful.



tional independence, such as for Figure 2.4(a):

$$P(C|A \wedge B) = P(C|B)$$

This means that the probability of  $C$ , given  $B$ , is exactly the same as the probability of  $C$ , given both  $B$  and  $A$ . Knowing that  $A$  has occurred doesn't make any difference to our beliefs about  $C$  if we already know that  $B$  has occurred. We also write this conditional independence as:  $A \perp\!\!\!\perp C|B$ .

In Figure 2.1(a), the probability that someone has dyspnoea depends directly only on whether they have cancer. If we don't know whether some woman has cancer, but we do find out she is a smoker, that would increase our belief both that she has cancer and that she suffers from shortness of breath. However, if we already *knew* she had cancer, then her smoking wouldn't make any difference to the probability of dyspnoea. That is, dyspnoea is conditionally independent of being a smoker *given* the patient has cancer.

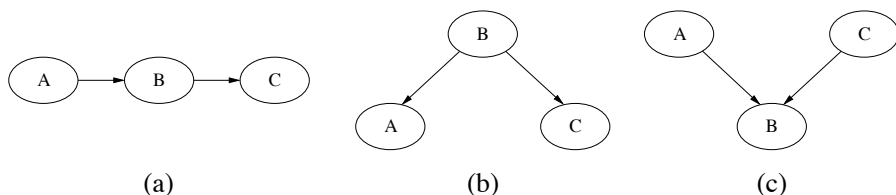


FIGURE 2.4: (a) Causal chain; (b) common cause; (c) common effect.

#### 2.4.4.2 Common causes

Two variables  $A$  and  $C$  having a common cause  $B$  is represented in Figure 2.4(b). In our example, cancer is a common cause of the two symptoms, a positive X-ray result and dyspnoea. Common causes (or common ancestors) give rise to the same conditional independence structure as chains:

$$P(C|A \wedge B) = P(C|B) \equiv A \perp\!\!\!\perp C|B$$

If there is no evidence or information about cancer, then learning that one symptom is present will increase the chances of cancer which in turn will increase the probability of the other symptom. However, if we already know about cancer, then an additional positive X-ray won't tell us anything new about the chances of dyspnoea.

#### 2.4.4.3 Common effects

A common effect is represented by a network v-structure, as in Figure 2.4(c). This represents the situation where a node (the effect) has two causes. Common effects (or their descendants) produce the exact opposite conditional independence structure

to that of chains and common causes. That is, the parents are marginally independent ( $A \perp\!\!\!\perp C$ ), but become dependent given information about the common effect (i.e., they are **conditionally dependent**):

$$P(A|C \wedge B) \neq P(A|B) \equiv A \not\perp\!\!\!\perp C|B$$

Thus, if we observe the effect (e.g., cancer), and then, say, we find out that one of the causes is absent (e.g., the patient does not smoke), this *raises* the probability of the other cause (e.g., that he lives in a polluted area) — which is just the inverse of explaining away.

### Compactness again

So we can now see *why* building networks with an order violating causal order can, and generally will, lead to additional complexity in the form of extra arcs. Consider just the subnetwork  $\{ \textit{Pollution}, \textit{Smoker}, \textit{Cancer} \}$  of Figure 2.1. If we build the subnetwork in that order we get the simple v-structure  $\textit{Pollution} \rightarrow \textit{Smoker} \leftarrow \textit{Cancer}$ . However, if we build it in the order  $\langle \textit{Cancer}, \textit{Pollution}, \textit{Smoker} \rangle$ , we will first get  $\textit{Cancer} \rightarrow \textit{Pollution}$ , because they are dependent. When we add *Smoker*, it will be dependent upon *Cancer*, because in reality there is a direct dependency there. But we shall also have to add a spurious arc to *Pollution*, because otherwise *Cancer* will act as a common cause, inducing a spurious dependency between *Smoker* and *Pollution*; the extra arc is necessary to reestablish marginal independence between the two.

### 2.4.5 d-separation

We have seen how Bayesian networks represent conditional independencies and how these independencies affect belief change during updating. The conditional independence in  $A \perp\!\!\!\perp C|B$  means that knowing the value of *B* **blocks** information about *C* being relevant to *A*, and vice versa. Or, in the case of Figure 2.4(c), *lack* of information about *B* blocks the relevance of *C* to *A*, whereas learning about *B* **activates** the relation between *C* and *A*.

These concepts apply not only between pairs of nodes, but also between sets of nodes. More generally, given the Markov property, it is possible to determine whether a set of nodes **X** is independent of another set **Y**, given a set of evidence nodes **E**. To do this, we introduce the notion of **d-separation** (from **direction-dependent separation**).

**Definition 2.1 Path (Undirected Path)** A **path** between two sets of nodes **X** and **Y** is any sequence of nodes between a member of **X** and a member of **Y** such that every adjacent pair of nodes is connected by an arc (regardless of direction) and no node appears in the sequence twice.

**Definition 2.2 Blocked path** A path is **blocked**, given a set of nodes  $\mathbf{E}$ , if there is a node  $Z$  on the path for which at least one of three conditions holds:

1.  $Z$  is in  $\mathbf{E}$  and  $Z$  has one arc on the path leading in and one arc out (chain).
2.  $Z$  is in  $\mathbf{E}$  and  $Z$  has both path arcs leading out (common cause).
3. Neither  $Z$  nor any descendant of  $Z$  is in  $\mathbf{E}$ , and both path arcs lead in to  $Z$  (common effect).

**Definition 2.3 d-separation** A set of nodes  $\mathbf{E}$  **d-separates** two other sets of nodes  $\mathbf{X}$  and  $\mathbf{Y}$  ( $\mathbf{X} \perp \mathbf{Y} \mid \mathbf{E}$ ) if every path from a node in  $\mathbf{X}$  to a node in  $\mathbf{Y}$  is **blocked** given  $\mathbf{E}$ .

If  $\mathbf{X}$  and  $\mathbf{Y}$  are **d-separated** by  $\mathbf{E}$ , then  $\mathbf{X}$  and  $\mathbf{Y}$  are **conditionally independent** given  $\mathbf{E}$  (given the Markov property). Examples of these three blocking situations are shown in Figure 2.5. Note that we have simplified by using single nodes rather than sets of nodes; also note that the evidence nodes  $\mathbf{E}$  are shaded.

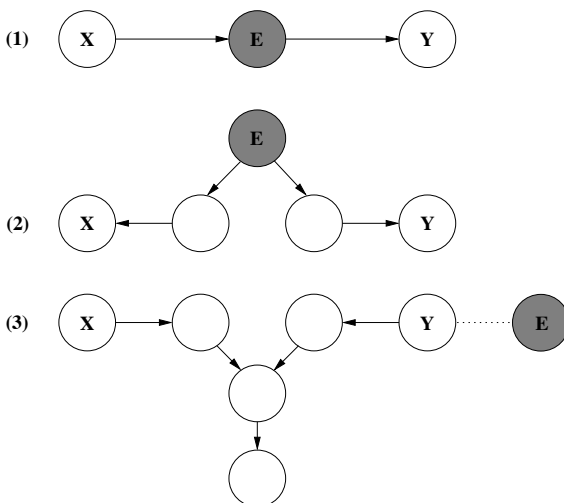


FIGURE 2.5: Examples of the three types of situations in which the path from  $X$  to  $Y$  can be blocked, given evidence  $E$ . In each case,  $X$  and  $Y$  are **d-separated** by  $E$ .

Consider d-separation in our cancer diagnosis example of Figure 2.1. Suppose an observation of the Cancer node is our evidence. Then:

1.  $P$  is d-separated from  $X$  and  $D$ . Likewise,  $S$  is d-separated from  $X$  and  $D$  (blocking condition 1).
2. While  $X$  is d-separated from  $D$  (condition 2).
3. However, if  $C$  had not been observed (and also not  $X$  or  $D$ ), then  $S$  would have been d-separated from  $P$  (condition 3).

**Definition 2.4 d-connection** Sets  $\mathbf{X}$  and  $\mathbf{Y}$  are **d-connected** given set  $\mathbf{E}$  ( $\mathbf{X} \not\perp \mathbf{Y} \mid \mathbf{E}$ ) if there is a path from a node in  $\mathbf{X}$  to a node in  $\mathbf{Y}$  which is not **blocked** given  $\mathbf{E}$ .

## 2.5 More examples

In this section we present further simple examples of BN modeling from the literature. We encourage the reader to work through these examples using BN software (see Appendix B).

### 2.5.1 Earthquake

**Example statement:** *You have a new burglar alarm installed. It reliably detects burglary, but also responds to minor earthquakes. Two neighbors, John and Mary, promise to call the police when they hear the alarm. John always calls when he hears the alarm, but sometimes confuses the alarm with the phone ringing and calls then also. On the other hand, Mary likes loud music and sometimes doesn't hear the alarm. Given evidence about who has and hasn't called, you'd like to estimate the probability of a burglary (from Pearl (1988)).*

A BN representation of this example is shown in Figure 2.6. All the nodes in this BN are Boolean, representing the true/false alternatives for the corresponding propositions. This BN models the assumptions that John and Mary do not perceive a burglary directly and they do not feel minor earthquakes. There is no explicit representation of loud music preventing Mary from hearing the alarm, nor of John's confusion of alarms and telephones; this information is summarized in the probabilities in the arcs from *Alarm* to *JohnCalls* and *MaryCalls*.

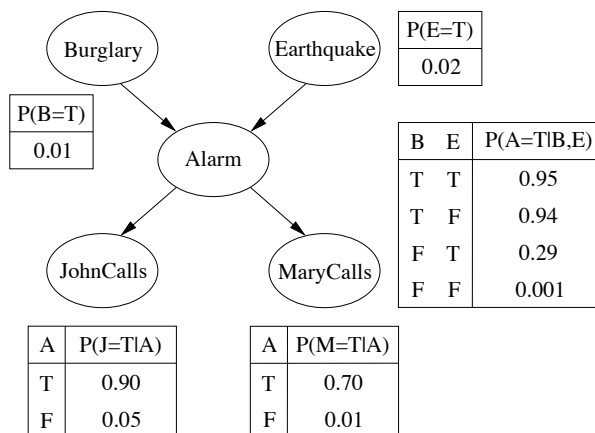


FIGURE 2.6: Pearl's Earthquake BN.

### 2.5.2 Metastatic cancer

**Example statement:** *Metastatic cancer is a possible cause of brain tumors and is also an explanation for increased total serum calcium. In turn, either of these could explain a patient falling into a coma. Severe headache is also associated with brain tumors. (This example has a long history in the literature, namely Cooper, 1984, Pearl, 1988, Spiegelhalter, 1986.)*

A BN representation of this metastatic cancer example is shown in Figure 2.7. All the nodes are Booleans. Note that this is a *graph*, not a tree, in that there is more than one path between the two nodes *M* and *C* (via *S* and *B*).

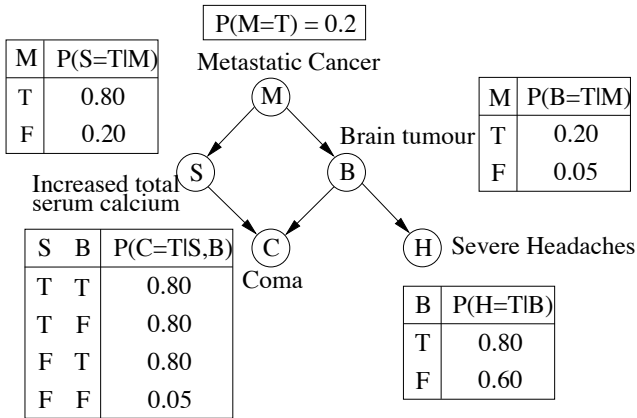


FIGURE 2.7: Metastatic cancer BN.

### 2.5.3 Asia

**Example Statement:** *Suppose that we wanted to expand our original medical diagnosis example to represent explicitly some other possible causes of shortness of breath, namely tuberculosis and bronchitis. Suppose also that whether the patient has recently visited Asia is also relevant, since TB is more prevalent there.*

Two alternative BN structures for the so-called Asia example are shown in Figure 2.8. In both networks all the nodes are Boolean. The left-hand network is based on the Asia network of Lauritzen and Spiegelhalter (1988). Note the slightly odd intermediate node *TBorC*, indicating that the patient has either tuberculosis or bronchitis. This node is not strictly necessary; however it reduces the number of arcs elsewhere, by summarizing the similarities between TB and lung cancer in terms of their relationship to positive X-ray results and dyspnoea. Without this node, as can be seen on the right, there are two parents for *X-ray* and three for *Dyspnoea*, with the same probabilities repeated in different parts of the CPT. The use of such an intermediate node is an example of “divorcing,” a model structuring method described in §10.3.6.

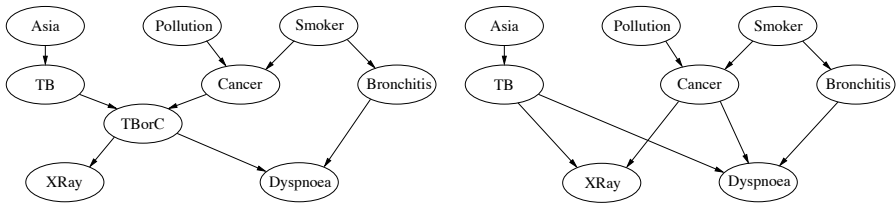


FIGURE 2.8: Alternative BNs for the "Asia" example.

---

## 2.6 Summary

Bayes' theorem allows us to update the probabilities of variables whose state has not been observed given some set of new observations. Bayesian networks automate this process, allowing reasoning to proceed in any direction across the network of variables. They do this by combining qualitative information about direct dependencies (perhaps causal relations) in arcs and quantitative information about the strengths of those dependencies in conditional probability distributions. Computational speed gains in updating accrue when the network is sparse, allowing d-separation to take advantage of conditional independencies in the domain (so long as the Markov property holds). Given a known set of conditional independencies, Pearl's network construction algorithm guarantees the development of a minimal network, without redundant arcs. In the next chapter, we turn to specifics about the algorithms used to update Bayesian networks.



---


## 2.7 Bibliographic notes

The text that marked the new era of Bayesian methods in artificial intelligence is Judea Pearl's *Probabilistic Reasoning in Intelligent Systems* (1988). This text played no small part in attracting the authors to the field, amongst many others. Richard Neapolitan's *Probabilistic Reasoning in Expert Systems* (1990) complements Pearl's book nicely, and it lays out the algorithms underlying the technology particularly well. Two more current introductions are Jensen and Nielsen's *Bayesian Networks and Decision Graphs* (2007), Kjærulff and Madsen's *Bayesian Networks and Influence Diagrams: A Guide to Construction and Analysis* (2008); both their level and treatment is similar to ours; however, they do not go as far with the machine learning and knowledge engineering issues we treat later. More technical discussions can be found in Cowell et al.'s *Probabilistic Networks and Expert Systems* (1999), Richard Neapolitan's *Learning Bayesian Networks* (2003) and Koller and Friedman's *Probabilistic Graphical Models: Principles and Techniques* (2009).

### A Quick Guide to Using BayesiaLab

**Installation:** Web Site [www.bayesia.com](http://www.bayesia.com). Download BayesiaLab zip file which is available for all platforms that support the Sun Java Runtime Environment (JRE) (Windows, Mac OS X, and Linux). This gives you a BayesiaLab.zip. Extract the contents of the zip file, to your computer's file system. The Sun Java Runtime environment is required to run BayesiaLab. The Sun JRE can be downloaded from the Sun Java Web site: [java.com](http://java.com) To run BayesiaLab, navigate to the installation directory on the command line, and run `java -Xms128M -Xmx512M -jar BayesiaLab.jar`


**Network Files:** BNs are stored in .xbl files, with icon . BayesiaLab comes with a Graphs folder of example networks. To open an existing network, select  or select Network→Open menu option.

**Evidence:** Evidence can only be added and removed in “Validation Mode”. To enter this mode either click on the  icon or click View→Validation Mode in the main menu.


To add evidence:



1. Double-click on the node for which you want to add evidence.
2. A “monitor” for the node will appear in the list in the right-hand portion of the BayesiaLab window. In the node's monitor, double-click on the variable, for which you would like to add evidence.

To remove evidence:

- In the node's monitor, double-click on the variable, for which you would like to remove evidence; or
- Click on  to remove all evidence (called “observations” in BayesiaLab).

**Editing/Creating a BN:** BNs can only be created or edited in “Modeling Mode”.

To enter this mode either click on the  icon or click View→Modeling Mode in the main menu. Note that BayesiaLab beliefs are given out of 100, not as direct probabilities (i.e. not numbers between 0 and 1).



- Add a node by selecting  and then left-clicking, onto the canvas where you want to place the node.
- Add an arc by selecting , then dragging the arc from the parent node to the child node.
- Double click on node, then click on the Probability Distribution tab to bring up the CPT. Entries can be added or changed by clicking on the particular cells.

**Saving a BN:** Select  or the Network→Save menu option.

FIGURE 2.9: A quick guide to using BayesiaLab.


### A Quick Guide to Using GeNIe

**Installation:** Web Site [www.genie.sis.pitt.edu](http://www.genie.sis.pitt.edu). Download GeNIe which is available for Windows. This gives you a `genie2_setup.exe`, an installer executable. Double-clicking the executable, will start the installation wizard.

**Network Files:** BNs are stored in `.xdsl` files, with icon . GeNIe comes with an `Examples` folder of example networks. To open an existing network, select  or select `File`→`Open Network` menu option, or double-click on the file.

**Compilation:** Once a GeNIe BN has been opened, before you can see the initial beliefs, you must first compile it:

- Click on ; or
- Select `Network`→`Update Beliefs` menu option.

Once the network is compiled, you can view the state of each node by hovering over the node's tick icon () , with your mouse.

**Evidence:** To add evidence:



- Left click on the node, and select `Node`→`Set Evidence` in GeNIe's menu system; or
- Right click on the node, and select `Set Evidence` in the right-click menu

To remove evidence:

- Right click on the node and select `Clear Evidence`; or
- Select `Network`→`Clear All Evidence` menu-option.

There is an option (`Network`→`Update Immediately`) to automatically recompile and update beliefs when new evidence is set.

**Editing/Creating a BN:** Double-clicking on a node will bring up a window showing node features.

- Add a node by selecting  and then “drag-and-drop” with the mouse, onto the canvas, or right-clicking on the canvas and then selecting `Insert Here`→`Change` from the menu.
- Add an arc by selecting , then left-click first on the parent node, then the child node.
- Double click on node, then click on the `Definition` tab to bring up the CPT. Entries can be added or changed by clicking on the particular cells.




**Saving a BN:** Select  or the `File`→`Save` menu option.


FIGURE 2.10: A quick guide to using GeNIe.





### A Quick Guide to Using Hugin



**Installation:** Web Site [www.hugin.com](http://www.hugin.com). Download Hugin Lite, which is available for MS Windows (95 / 98 / NT4 / 2000 / XP), Solaris Sparc, Solaris X86 and Linux. This gives you `HuginLite63.exe`, a self-extracting zip archive. Double-clicking will start the extraction process.





**Network Files:** BNs are stored in `.net` files, with icon . Hugin comes with a `samples` folder of example networks. To open an existing network, select , or select `File`→`Open` menu option, or double-click on the file.


**Compilation:** Once a Hugin BN has been opened, before you can see the initial beliefs or add evidence, you must first compile it (which they call “switch to run mode”): click on , or select `Network`→`Run(in edit mode)`, or `Recompile (in run mode)` menu option.

This causes another window to appear on the left side of the display (called the Node Pane List), showing the network name, and all the node names. You can display/hide the states and beliefs in several ways. You can select a particular node by clicking on the ‘+’ by the node name, or all nodes with `View`→`Expand Node List`, or using icon . Unselecting is done similarly with ‘-’, or `View`→`Collapse Node List`, or using icon .

Selecting a node means all its states will be displayed, together with a bar and numbers showing the beliefs. Note that Hugin beliefs are given as percentages out of 100, not as direct probabilities (i.e., not numbers between 0 and 1).

**Editing/Creating a BN:** You can only change a BN when you are in “edit” mode, which you can enter by selecting the edit mode icon , or selecting `Network`→`Edit`. Double-clicking on a node will bring up a window showing node features, or use icon .

- Add a node by selecting either  (for discrete node) or  (for continuous node), `Edit`→`Discrete Chance Tool` or `Edit`→`Continuous Chance Tool`. In each case, you then “drag-and-drop” with the mouse.
- Add an arc by selecting either , or `Edit`→`Link Tool`, then left-click first on the parent node, then the child node.
- Click on the , icon to split the window horizontally between a Tables Pane (above), showing the CPT of the currently selected node, and the network structure (below).


**Saving a BN:** Select , or the `File`→`Save` menu option. Note that the Hugin Lite demonstration version limits you to networks with up to 50 nodes and learn from maximum 500 cases; for larger networks, you need to buy a license.


**Junction trees:** To change the triangulation method select `Network`→`Network Properties`→`Compilation`, then turn on “Specify Triangulation Method.” To view, select the `Show Junction Tree` option.

FIGURE 2.11: A quick guide to using Hugin.


### A Quick Guide to Using Netica

**Installation:** Web Site [www.norsys.com](http://www.norsys.com). Download Netica, which is available for MS Windows (95 / 98 / NT4 / 2000 / XP / Vista), and MacIntosh OSX. This gives you *Netica\_Win.exe*, a self-extracting zip archive. Double-clicking will start the extraction process.

**Network Files:** BNs are stored in *.dne* files, with icon . Netica comes with a folder of example networks, plus a folder of tutorial examples. To open an existing network:

- Select 
- Select *File*→*Open* menu option; or
- Double-click on the BN *.dne* file.

**Compilation:** Once a Netica BN has been opened, before you can see the initial beliefs or add evidence, you must first compile it:


- Click on 
- Select *Network*→*Compile* menu option.

Once the network is compiled, numbers and bars will appear for each node state. Note that Netica beliefs are given out of 100, not as direct probabilities (i.e., not numbers between 0 and 1).

**Evidence:** To add evidence:



- Left-click on the node state name; or
- Right-click on node and select particular state name.

To remove evidence:

- Right-click on node and select *unknown*; or
- Select ; or
- Select *Network*→*Remove findings* menu option.

There is an option (*Network*→*Automatic Update*) to automatically re-compile and update beliefs when new evidence is set.

**Editing/Creating a BN:** Double-clicking on a node will bring up a window showing node features.

- Add a node by selecting either ; or *Modify*→*Add nature node*, then “drag-and-drop” with the mouse.
- Add an arc by selecting either ; or *Modify*→*Add link*, then left-click first on the parent node, then the child node.
- Double-click on node, then click on the *Table* button to bring up the CPT. Entries can be added or changed by clicking on the particular cells.


**Saving a BN:** Select  or the *File*→*Save* menu option. Note that the Netica Demonstration version only allows you to save networks with up to 15 nodes. For larger networks, you need to buy a license.

FIGURE 2.12: A quick guide to using Netica.

## 2.8 Problems

### Modeling

These modeling exercises should be done using a BN software package (see our **Quick Guides to Using Netica** in Figure 2.12, **Hugin** in Figure 2.11, **GeNIe** in Figure 2.10, or **BayesiaLab** in Figure 2.9, and also Appendix B).

Also note that various information, including Bayesian network examples in Netica's .dne format, can be found at the book Web site:

<http://www.csse.monash.edu.au/bai>

### Problem 1

Construct a network in which explaining away operates, for example, incorporating multiple diseases sharing a symptom. Operate and demonstrate the effect of explaining away. *Must* one cause explain away the other? Or, can the network be parameterized so that this doesn't happen?

### Problem 2

"Fred's LISP dilemma." *Fred is debugging a LISP program. He just typed an expression to the LISP interpreter and now it will not respond to any further typing. He can't see the visual prompt that usually indicates the interpreter is waiting for further input. As far as Fred knows, there are only two situations that could cause the LISP interpreter to stop running: (1) there are problems with the computer hardware; (2) there is a bug in Fred's code. Fred is also running an editor in which he is writing and editing his LISP code; if the hardware is functioning properly, then the text editor should still be running. And if the editor is running, the editor's cursor should be flashing. Additional information is that the hardware is pretty reliable, and is OK about 99% of the time, whereas Fred's LISP code is often buggy, say 40% of the time.*<sup>5</sup>

1. Construct a Belief Network to represent and draw inferences about Fred's dilemma.

First decide what your domain variables are; these will be your network nodes. Hint: 5 or 6 Boolean variables should be sufficient. Then decide what the causal relationships are between the domain variables and add directed arcs in the network from cause to effect. Finally, you have to add the conditional probabilities for nodes that have parents, and the prior probabilities for nodes without parents. Use the information about the hardware reliability and how often Fred's code is buggy. Other probabilities haven't been given to you explicitly; choose values that seem reasonable and explain why in your documentation.

---

<sup>5</sup>Based on an example used in Dean, T., Allen, J. and Aloimonos, Y. *Artificial Intelligence Theory and Practice* (Chapter 8), Benjamin/Cumming Publishers, Redwood City, CA. 1995.

2. Show the belief of each variable before adding any evidence, i.e., about the LISP visual prompt not being displayed.
3. Add the evidence about the LISP visual prompt not being displayed. After doing belief updating on the network, what is Fred's belief that he has a bug in his code?
4. Suppose that Fred checks the screen and the editor's cursor is still flashing. What effect does this have on his belief that the LISP interpreter is misbehaving because of a bug in his code? Explain the change in terms of diagnostic and predictive reasoning.

### Problem 3

*“A Lecturer’s Life.” Dr. Ann Nicholson spends 60% of her work time in her office. The rest of her work time is spent elsewhere. When Ann is in her office, half the time her light is off (when she is trying to hide from students and get research done). When she is not in her office, she leaves her light on only 5% of the time. 80% of the time she is in her office, Ann is logged onto the computer. Because she sometimes logs onto the computer from home, 10% of the time she is not in her office, she is still logged onto the computer.*

1. Construct a Bayesian network to represent the “Lecturer’s Life” scenario just described.
2. Suppose a student checks Dr. Nicholson’s login status and sees that she is logged on. What effect does this have on the student’s belief that Dr. Nicholson’s light is on?

### Problem 4

*“Jason the Juggler.” Jason, the robot juggler, drops balls quite often when its battery is low. In previous trials, it has been determined that when its battery is low it will drop the ball 9 times out of 10. On the other hand when its battery is not low, the chance that it drops a ball is much lower, about 1 in 100. The battery was recharged recently, so there is only a 5% chance that the battery is low. Another robot, Olga the observer, reports on whether or not Jason has dropped the ball. Unfortunately Olga’s vision system is somewhat unreliable. Based on information from Olga, the task is to represent and draw inferences about whether the battery is low depending on how well Jason is juggling.<sup>6</sup>*

1. Construct a Bayesian network to represent the problem.
2. Which probability tables show where the information on how Jason’s success is related to the battery level, and Olga’s observational accuracy, are encoded in the network?

---

<sup>6</sup>Variation of Exercise 19.6 in Nilsson, N.J. *Artificial Intelligence: A New Synthesis*, Copyright (1998). With permission from Elsevier.

3. Suppose that Olga reports that Jason has dropped the ball. What effect does this have on your belief that the battery is low? What type of reasoning is being done?

### Problem 5

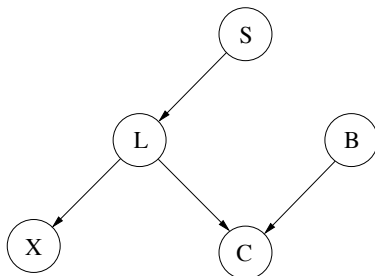
Come up with your own problem involving reasoning with evidence and uncertainty. Write down a text description of the problem, then model it using a Bayesian network. Make the problem sufficiently complex that your network has at least 8 nodes and is multiply-connected (i.e., not a tree or a polytree).

1. Show the beliefs for each node in the network before any evidence is added.
2. Which nodes are d-separated with no evidence added?
3. Which nodes in your network would be considered evidence (or observation) nodes? Which might be considered the query nodes? (Obviously this depends on the domain and how you might use the network.)
4. Show how the beliefs change in a form of diagnostic reasoning when evidence about at least one of the domain variables is added. Which nodes are d-separated with this evidence added?
5. Show how the beliefs change in a form of predictive reasoning when evidence about at least one of the domain variables is added. Which nodes are d-separated with this evidence added?
6. Show how the beliefs change through “explaining away” when particular combinations of evidence are added.
7. Show how the beliefs change when you change the priors for a root node (rather than adding evidence).

## Conditional Independence

### Problem 6

Consider the following Bayesian network for another version of the medical diagnosis example, where  $B$ =*Bronchitis*,  $S$ =*Smoker*,  $C$ =*Cough*,  $X$ =*Positive X-ray* and  $L$ =*Lung cancer* and all nodes are Booleans.



List the pairs of nodes that are conditionally independent in the following situations:

1. There is no evidence for any of the nodes.

2. The cancer node is set to true (and there is no other evidence).
3. The smoker node is set to true (and there is no other evidence).
4. The cough node is set to true (and there is no other evidence).

## Variable Ordering

### Problem 7

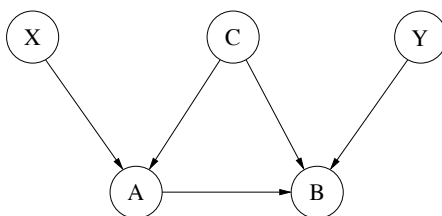
Consider the Bayesian network given for the previous problem.

1. What variable ordering(s) could have been used to produce the above network using the network construction algorithm (Algorithm 2.1)?
2. Given different variable orderings, what network structure would result from this algorithm? Use only pen and paper for now! Compare the number of parameters required by the CPTs for each network.

## d-separation

### Problem 8

Consider the following graph.



1. Find all the sets of nodes that d-separate X and Y (not including either X or Y in such sets).
2. Try to come up with a real-world scenario that might be modeled with such a network structure.

### Problem 9

Design an internal representation for a Bayesian network structure; that is, a representation for the nodes and arcs of a Bayesian network (but not necessarily the parameters — prior probabilities and conditional probability tables). Implement a function which generates such a data structure from the Bayesian network described by a Netica `dne` input file. Use this function in the subsequent problems. (Sample `dne` files are available from the book Web site.)

### Problem 10

Implement the network construction algorithm (Algorithm 2.1). Your program should take as input an ordered list of variables and prompt for additional input from

the keyboard about the conditional independence of variables as required. It should generate a Bayesian network in the internal representation designed above. It should also print the network in some human-readable form.

### Problem 11

Given as input the internal Bayesian network structure  $N$  (in the representation you have designed above), write a function which returns all undirected paths (Definition 2.1) between two sets  $X$  and  $Y$  of nodes in  $N$ .

Test your algorithm on various networks, including at least

- The d-separation network example from Problem 8, `dsepEg.dne`
- `Cancer_Neapolitan.dne`
- `ALARM.dne`

Summarize the results of these experiments.

### Problem 12

Given the internal Bayesian network structure  $N$ , implement a **d-separation oracle** which, for any three sets of nodes input to it,  $X$ ,  $Y$ , and  $Z$ , returns:

- **true** if  $X \perp Y | Z$  (i.e.,  $Z$  d-separates  $X$  and  $Y$  in  $N$ );
- **false** if  $X \not\perp Y | Z$  (i.e.,  $X$  and  $Y$  given  $Z$  are d-connected in  $N$ );
- some diagnostic (a value other than **true** or **false**) if an error in  $N$  is encountered.

Run your algorithm on a set of test networks, including at least the three network specified for Problem 11. Summarize the results of these experiments.

### Problem 13

Modify your network construction algorithm from Problem 9 above to use the d-separation oracle from the last problem, instead of input from the user. Your new algorithm should produce exactly the same network as that used by the oracle whenever the variable ordering provided it is compatible with the oracle's network. Experiment with different variable orderings. Is it possible to generate a network which is simpler than the oracle's network?