[Unit-5: Knowledge Representation]

Introduction to Artificial Intelligence (CSC-355)

Bal Krishna Subedi

Knowledge Representation

Knowledge:

Knowledge is a theoretical or practical understanding of a subject or a domain. Knowledge is also the sum of what is currently known.

Knowledge is ―the sum of what is known: the body of truth, information, and principles acquired by mankind.‖ Or, "Knowledge is what I know, Information is what we know."

There are many other definitions such as:

- Knowledge is "information combined with experience, context, interpretation, and reflection. It is a high-value form of information that is ready to apply to decisions and actions." (T. Davenport et al., 1998)

- Knowledge is ―human expertise stored in a person‘s mind, gained through experience, and interaction with the person‘s environment." (Sunasee and Sewery, 2002)

- Knowledge is ―information evaluated and organized by the human mind so that it can be used purposefully, e.g., conclusions or explanations." (Rousa, 2002)

Knowledge consists of information that has been:

- interpreted,
- categorised,
- applied, experienced and revised.

In general, knowledge is more than just data, it consist of: facts, ideas, beliefs, heuristics, associations, rules, abstractions, relationships, customs.

Research literature classifies knowledge as follows:

Classification-based Knowledge	»	Ability to classify information
Decision-oriented Knowledge	»	Choosing the best option
Descriptive knowledge	»	State of some world (heuristic)
Procedural knowledge	»	How to do something
Reasoning knowledge	»	What conclusion is valid in what situation?
Assimilative knowledge	»	What its impact is?

Knowledge Representation

Knowledge representation (KR) is the study of how knowledge about the world can be represented and what kinds of reasoning can be done with that knowledge. Knowledge Representation is the method used to encode knowledge in Intelligent Systems.

Since knowledge is used to achieve intelligent behavior, the fundamental goal of knowledge representation is to represent knowledge in a manner as to facilitate inferencing (i.e. drawing conclusions) from knowledge. A successful representation of some knowledge must, then, be in a form that is *understandable* by humans, and must cause the system using the knowledge to *behave* as if it knows it.

Some issues that arise in knowledge representation from an AI perspective are:

- How do people represent knowledge?
- What is the nature of knowledge and how do we represent it?
- Should a representation scheme deal with a particular domain or should it be general purpose?
- How expressive is a representation scheme or formal language?
- Should the scheme be declarative or procedural?

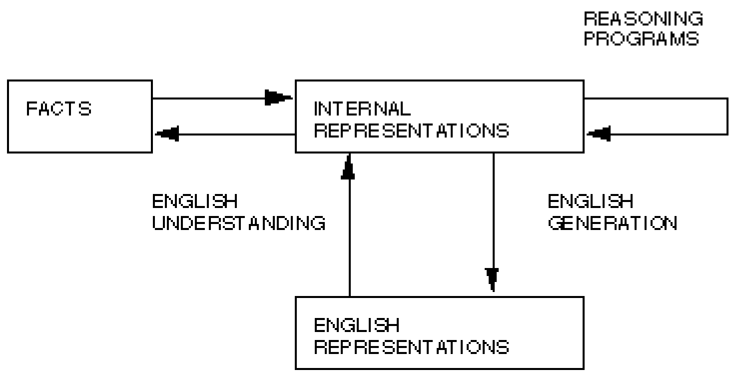


Fig: Two entities in Knowledge Representaion

For example: English or natural language is an obvious way of representing and handling facts. Logic enables us to consider the following fact: *spot is a dog* as *dog(spot)* We could then infer that all dogs have tails with: $\forall x$: *dog(x)* $\rightarrow$*hasatail(x)* We can then deduce:

hasatail(Spot)

Using an appropriate backward mapping function the English sentence *Spot has a tail can be generated.*

<u>Properties for Knowledge Representation Systems</u>

The following properties should be possessed by a knowledge representation system.

Representational Adequacy
- the ability to represent the required knowledge;

Inferential Adequacy
- the ability to manipulate the knowledge represented to produce new knowledge corresponding to that inferred from the original;

Inferential Efficiency

- the ability to direct the inferential mechanisms into the most productive directions by storing appropriate guides;

Acquisitional Efficiency

- the ability to acquire new knowledge using automatic methods wherever possible rather than reliance on human intervention.

Formal logic-connectives:

In logic, a **logical connective** (also called a **logical operator**) is a symbol or word used to connect two or more sentences (of either a formal or a natural language) in a grammatically valid way, such that the compound sentence produced has a truth value dependent on the respective truth values of the original sentences.

Each logical connective can be expressed as a function, called a truth function. For this reason, logical connectives are sometimes called **truth-functional connectives**.

Commonly used logical connectives include:

- Negation (not) (¬ or ~)
- Conjunction (and) ($\wedge$, &, or ·)
- Disjunction (or) ($\vee$ or $\vee$)
- Material implication (if...then) ($\longrightarrow$, $\Rightarrow$ or $\supset$)
- Biconditional (if and only if) (iff) (xnor) ($\longleftrightarrow$, $\equiv$, or =)

For example, the meaning of the statements *it is raining* and *I am indoors* is transformed when the two are combined with logical connectives:

- It is raining **and** I am indoors ($P \wedge Q$)
- **If** it is raining, **then** I am indoors ($P \longrightarrow Q$)
- It is raining **if** I am indoors ($Q \longrightarrow P$)
- It is raining **if and only if** I am indoors ($P \longleftrightarrow Q$)
- It is **not** raining ($\neg P$)

For statement *P* = *It is raining* and *Q* = *I am indoors*.

Truth Table:

A proposition in general contains a number of variables. For example (P $\vee$Q) contains variables P and Q each of which represents an arbitrary proposition. Thus a proposition takes different values depending on the values of the constituent variables. This relationship of the value of a proposition and those of its constituent variables can be represented by a table. It tabulates the value of a proposition for all possible values of its variables and it is called a truth table.

For example the following table shows the relationship between the values of P, Q and P $\vee$Q:

OR		
P	**Q**	**(P ⋁Q)**
F	F	F
F	T	T
T	F	T
T	T	T

Logic:

Logic is a formal language for representing knowledge such that conclusions can be drawn. **Logic** makes statements about the world which are true (or false) if the state of affairs it represents is the case (or not the case). Compared to natural languages (expressive but context sensitive) and programming languages (good for concrete data structures but not expressive) logic combines the advantages of natural languages and formal languages. Logic is concise, unambiguous, expressive, context insensitive, effective for inferences.

It has syntax, semantics, and proof theory.

Syntax: Describe possible configurations that constitute sentences.

Semantics: Determines what fact in the world, the sentence refers to i.e. the interpretation. Each sentence make claim about the world (meaning of sentence).Semantic property include truth and falsity.

Syntax is concerned with the rules used for constructing, or transforming the symbols and words of a language, as contrasted with the semantics of a language which is concerned with its meaning.

Proof theory (Inference method): set of rules for generating new sentences that are necessarily true given that the old sentences are true.

We will consider two kinds of logic: **propositional logic** and **first-order logic** or more precisely first-order **predicate calculus**. Propositional logic is of limited expressiveness but is useful to introduce many of the concepts of logic's syntax, semantics and inference procedures.

Entailment:

Entailment means that one thing follows from another:

$KB \models \alpha$

Knowledge base KB entails sentence α if and only if α is true in all worlds where KB is true

E.g., x + y =4 entails 4=x + y

Entailment is a relationship between sentences (i.e., syntax) that is based on semantics.

We can determine whether S |= P by finding Truth Table for S and P, if any row of Truth Table where all formulae in S is true.

Example:

P	$P \to Q$	Q
True	True	True
True	False	False
False	True	True
False	True	False

Therefore {P, P→Q} |= Q. Here, only row where both P and P→Q are True, Q is also True. Here, S= (P, P→Q} and P= {Q}.

Models

Logicians typically think in terms of models, in place of ―possible world‖, which are formally structured worlds with respect to which truth can be evaluated.

m is a model of a sentence α if α is true in m.

M(α) is the set of all models of α.

Tautology:

A formula of propositional logic is a **tautology** if the formula itself is always true regardless of which valuation is used for the propositional variables.

There are infinitely many tautologies. Examples include:

- $(A \lor \neg A)$("*A* or not *A*"), the law of the excluded middle. This formula has only one propositional variable, *A*. Any valuation for this formula must, by definition, assign *A* one of the truth values *true* or *false*, and assign ¬*A* the other truth value.
- $(A \to B) \Leftrightarrow (\neg B \to \neg A)$("if *A* implies *B* then not-*B* implies not-*A*", and vice versa), which expresses the law of contraposition.
- $((A \to B) \land (B \to C)) \to (A \to C)$("if *A* implies *B* and *B* implies *C*, then *A* implies *C*"), which is the principle known as syllogism.

The definition of ***tautology*** can be extended to sentences in predicate logic, which may contain quantifiers, unlike sentences of propositional logic. In propositional logic, there is no distinction between a tautology and a **logically valid formula**. In the context of predicate logic, many authors define a tautology to be a sentence that can be obtained by taking a tautology of propositional logic and uniformly replacing each propositional variable by a first-order formula (one formula per propositional variable). The set of such formulas is a proper subset of the set of logically valid sentences of predicate logic (which are the sentences that are true in every model).

There are also propositions that are always false such as (P $\wedge\neg$P). Such a proposition is called a **contradiction**.

A proposition that is neither a tautology nor a contradiction is called a **contingency**. For example (P $\vee$Q) is a contingency.

Validity:

The term **validity** in logic (also **logical validity**) is largely synonymous with logical truth, however the term is used in different contexts. Validity is a property of formulae, statements and arguments**. A logically valid argument is one where the conclusion follows from the premises. An invalid argument is where the conclusion does not follow from the premises.** A formula of a formal language is a valid formula if and only if it is true under every possible interpretation of the language.

Saying that an argument is valid is equivalent to saying that it is logically impossible that the premises of the argument are true and the conclusion false. A less precise but intuitively clear way of putting this is to say that in a valid argument IF the premises are true, then the conclusion must be true.

An argument that is not valid is said to be ―invalid‖.

An example of a valid argument is given by the following well-known syllogism:

All men are mortal.
Socrates is a man.
Therefore, Socrates is mortal.

What makes this a valid argument is not that it has true premises and a true conclusion, but the logical necessity of the conclusion, given the two premises.

The following argument is of the same logical form but with false premises and a false conclusion, and it is equally valid:

All women are cats.
All cats are men.
Therefore, all women are men.

This argument has false premises and a false conclusion. This brings out the hypothetical character of validity. What the validity of these arguments amounts to, is that it assures us the conclusion must be true IF the premises are true.

Thus, an argument is valid if the premises and conclusion follow a logical form. This essentially means that the conclusion logically follows from the premises. An argument is valid if and only if the truth of its premises entails the truth of its conclusion. It would be self-contradictory to affirm the premises and deny the conclusion

Deductive Reasoning:

Deductive reasoning, also called **Deductive logic**, is reasoning which constructs or evaluates deductive arguments. Deductive arguments are attempts to show that a conclusion necessarily follows from a set of premises. **A deductive argument is valid if the conclusion does follow necessarily from the premises, i.e., if the conclusion must be true provided that the premises are true**. A deductive argument is sound if it is valid AND its premises are true. Deductive arguments are valid or invalid, sound or unsound, but are never false or true.

An example of a deductive argument:

1. All men are mortal
2. Socrates is a man
3. Therefore, Socrates is mortal

The first premise states that all objects classified as 'men' have the attribute 'mortal'. The second premise states that 'Socrates' is classified as a man- a member of the set 'men'. The conclusion states that 'Socrates' must be mortal because he inherits this attribute from his classification as a man.

Deductive arguments are generally evaluated in terms of their *validity* and *soundness*. An argument is *valid* if it is impossible both for its premises to be true and its conclusion to be false. An argument can be valid even though the premises are false.

This is an example of a valid argument. The first premise is false, yet the conclusion is still valid.

All fire-breathing rabbits live on Mars
All humans are fire-breathing rabbits
Therefore, all humans live on Mars

This argument is valid but not *sound* In order for a deductive argument to be sound, the deduction must be valid and the premise must **all** be true.

Let‘s take one of the above examples.

1. All monkeys are primates
2. All primates are mammals
3. All monkeys are mammals

This is a sound argument because it is actually true in the real world. The premises are true and so is the conclusion. They logically follow from one another to form a concrete argument that can't be denied. Where validity doesn't have to do with the actual truthfulness of an argument, soundness does.

A theory of deductive reasoning known as categorical or term logic was developed by Aristotle, but was superseded by propositional (sentential) logic and predicate logic.

Deductive reasoning can be contrasted with inductive reasoning. In cases of inductive reasoning, it is possible for the conclusion to be false even though the premises are true and the argument's form is cogent.

Well Formed Formula: (wff)

It is a syntactic object that can be given a semantic meaning. A formal language can be considered to be identical to the set containing all and only its wffs.

A key use of wffs is in propositional logic and predicate logics such as first-order logic. In those contexts, a formula is a string of symbols φ for which it makes sense to ask "is φ true?", once any free variables in φ have been instantiated. In formal logic, proofs can be represented by sequences of wffs with certain properties, and the final wff in the sequence is what is proven.

The well-formed formulas of **propositional calculus** are expressions such as $(A \wedge (B \vee C))$Their definition begins with the arbitrary choice of a set *V* of propositional variables. The alphabet consists of the letters in *V* along with the symbols for the propositional connectives and parentheses "(" and ")", all of which are assumed to not be in *V*. The wffs will be certain expressions (that is, strings of symbols) over this alphabet.

The well-formed formulas are inductively defined as follows:

- Each propositional variable is, on its own, a wff.
- If φ is a wff, then ¬φ is a wff.
- If φ and ψ are wffs, and • is any binary connective, then (φ • ψ) is a wff. Here • could be ∨, ∧, →, or ↔.

The WFF for **predicate calculus** is defined to be the smallest set containing the set of atomic WFFs such that the following holds:

1. $\neg\phi$is a WFF when $\phi$is a WFF
2. $(\phi \wedge \psi)$and $(\phi \vee \psi)$are WFFs when $\phi$and $\psi$are WFFs;
3. $\exists x\, \phi$is a WFF when *x* is a variable and $\phi$is a WFF;
4. $\forall x\, \phi$ is a WFF when $x$is a variable and $\phi$is a WFF (alternatively, $\forall x\, \phi$ could be defined as an abbreviation for $\neg\exists x\, \neg\phi$).

If a formula has no occurrences of $\exists x$ or $\forall x$, for any variable x, then it is called *quantifier-free*. An *existential formula* is a string of existential quantification followed by a quantifier-free formula.

Propositional Logic:

Propositional logic represents knowledge/ information in terms of propositions. Prepositions are facts and non-facts that can be true or false. Propositions are expressed using ordinary declarative sentences. Propositional logic is the simplest logic.

Syntax:

The syntax of propositional logic defines the allowable sentences. The atomic sentences- the indivisible syntactic elements- consist of single proposition symbol. Each such symbol stands for a proposition that can be true or false. We use the symbols like P1, P2 to represent sentences.

The complex sentences are constructed from simpler sentences using logical connectives. There are five connectives in common use:

¬ (*negation*), ^ (*conjunction*), ∨ (*disjunction*), ⇒ (*implication*), ⇔ (*biconditional*)

The order of precedence in propositional logic is from (highest to lowest): ¬ , ^ , ∨, ⇒, ⇔.

Propositional logic is defined as:

If S is a sentence, ¬S is a sentence (*negation*)
If S1 and S2 are sentences, S1 ^ S2 is a sentence (*conjunction*)
If S1 and S2 are sentences, S1 ∨ S2 is a sentence (*disjunction*)
If S1 and S2 are sentences, S1 ⇒ S2 is a sentence (*implication*)
If S1 and S2 are sentences, S1 ⇔ S2 is a sentence (*biconditional*)

Formal grammar for propositional logic can be given as below:

Sentence → AutomicSentence | ComplexSentence
AutomicSentence → True | False | Symbol
Symbol → P | Q | R …………
ComplexSentence → ¬Sentence
| (Sentence ^ Sentence)
| (Sentence ∨ Sentence)
| (Sentence ⇒ Sentence)
| (Sentence ⇔ Sentence)

Semantics:

Each model specifies true/false for each proposition symbol

Rules for evaluating truth with respect to a model:

$\neg$S is true if, S is false
S1 ^ S2 is true if, S1 is true and S2 is true
S1 $\vee$ S2 is true if, S1 is true or S2 is true
S1 $\Rightarrow$ S2 is true if, S1 is false or S2 is true
S1 $\Leftrightarrow$ S2 is true if, S1 $\Rightarrow$ S2 is true and S2 $\Rightarrow$ S1 is true

Truth Table showing the evaluation of semantics of complex sentences:

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

Logical equivalence:

Two sentences α and ß are *logically equivalent* ($\alpha \equiv$ ß) iff true they are true inn same set of models or Two sentences α and ß are *logically equivalent* ($\alpha \equiv$ ß) iff α |= ß and ß |= α.

$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$ commutativity of $\wedge$
$(\alpha \vee \beta) \equiv (\beta \vee \alpha)$ commutativity of $\vee$
$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma))$ associativity of $\wedge$
$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma))$ associativity of $\vee$
$\neg(\neg\alpha) \equiv \alpha$ double-negation elimination
$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha)$ contraposition
$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$ implication elimination
$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$ biconditional elimination
$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$ de Morgan
$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$ de Morgan
$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$ distributivity of $\wedge$ over $\vee$
$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$ distributivity of $\vee$ over $\wedge$

Validity:

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

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

Valid sentences are also known as tautologies. Every valid sentence is logically equivalent to True

Satisfiability:

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$\neg\wedge$A

Validity and satisfiablity are related concepts

- α is valid iff $\neg\alpha$ is unsatisfiable
- α is satisfiable iff $\neg\alpha$ is not valid

Satisfiability is connected to inference via the following:

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

Inference rules in Propositional Logic

Modus Ponens

$$\frac{\alpha \Rightarrow \beta, \alpha}{\beta}$$

And-elimination

$$\frac{\alpha \wedge \beta}{\alpha}$$

Monotonicity: the set of entailed sentences can only increase as information is added to the knowledge base.

For any sentence α and β if KB $\models \alpha$ then KB $\wedge \beta \models \alpha$.

Resolution

Unit resolution rule:

Unit resolution rule takes a clause – a disjunction of literals – and a literal and produces a new clause. Single literal is also called unit clause.

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k}$$

Where ℓ_i and m are complementary literals

Generalized resolution rule:

Generalized resolution rule takes two clauses of any length and produces a new clause as below.

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

For example:

$$\frac{\ell_1 \vee \ell_2, \qquad \neg\ell_2 \vee \ell_3}{\ell_1 \vee \ell_3}$$

Resolution Uses CNF (Conjunctive normal form)

- **Conjunction of disjunctions of literals (clauses)**

The resolution rule is sound:

- Only entailed sentences are derived

Resolution is complete in the sense that it can always be used to either confirm or refute a sentence (it can not be used to enumerate true sentences.)

<u>Conversion to CNF:</u>

A sentence that is expressed as a conjunction of disjunctions of literals is said to be in conjunctive normal form (CNF). A sentence in CNF that contains only k literals per clause is said to be in k-CNF.

<u>Algorithm:</u>

Eliminate ↔rewriting P↔Q as (P→Q)∧(Q→P)

Eliminate →rewriting P→Q as ¬P∨Q

Use De Morgan‘s laws to push ¬ inwards:

- rewrite ¬(P∧Q) as ¬P∨¬Q

- rewrite ¬(P∨Q) as ¬P∧¬Q

Eliminate double negations: rewrite ¬¬*P* as *P*

Use the distributive laws to get CNF:

- rewrite (P∧Q)∨R as (P∨R)∧(Q∨R)

Flatten nested clauses:

- (P∧Q) ∧ R as P∧Q ∧ R

- (P∨Q)∨R as P∨Q∨R

<u>Example</u>: Let‘s illustrate the conversion to CNF by using an example.

B ⇔ (A ∨ C)

- Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow ß$ with $(\alpha \Rightarrow ß)\wedge(ß \Rightarrow \alpha)$.
  - $(B \Rightarrow (A \vee C)) \wedge ((A \vee C) \Rightarrow B)$

- Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow ß$ with $\neg \alpha \vee ß$.
  - $(\neg B \vee A \vee C) \wedge (\neg(A \vee C) \vee B)$

- Move $\neg$ inwards using de Morgan's rules and double-negation:
  - $(\neg B \vee A \vee C) \wedge ((\neg A \wedge \neg C) \vee B)$

- Apply distributivity law ($\wedge$ over $\vee$) and flatten:
  - $(\neg B \vee A \vee C) \wedge (\neg A \vee B) \wedge (\neg C \vee B)$

Resolution algorithm

- Convert KB into CNF
- Add negation of sentence to be entailed into KB i.e. $(KB \wedge \neg\alpha)$
- Then apply resolution rule to resulting clauses.
- The process continues until:
  - There are no new clauses that can be added
    Hence *KB* **does not** entail α
  - Two clauses resolve to entail the empty clause.
    Hence *KB* **does** entail α

Example: Consider the knowledge base given as: $KB = (B \Leftrightarrow (A\vee C)) \wedge \neg B$
Prove that $\neg A$ can be inferred from above KB by using resolution.

Solution:

At first, convert KB into CNF

$B \Rightarrow (A \vee C)) \wedge ((A \vee C) \Rightarrow B) \wedge\neg B$

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

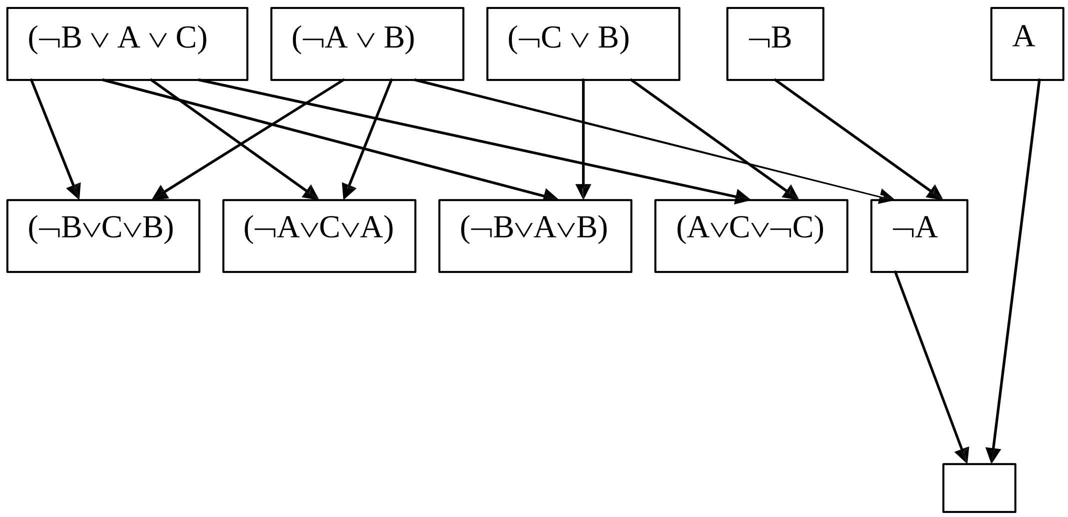
$(\neg B \vee A \vee C) \wedge ((\neg A \wedge \neg C) \vee B) \wedge\neg B$

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

Add negation of sentence to be inferred from KB into KB

Now KB contains following sentences all in CNF
$(\neg B \vee A \vee C)$
$(\neg A \vee B)$
$(\neg C \vee B)$
$\neg B$
A (negation of conclusion to be proved)

Now use Resolution algorithm



Resolution: More Examples

1. KB= $\{(G \vee H) \rightarrow (\neg J \wedge \neg K), G\}$. Show that KB $\vdash \neg J$

   Solution:

   Clausal form of $(G \vee H) \rightarrow (\neg J \wedge \neg K)$ is

   $\{\neg G \vee \neg J, \neg H \vee \neg J, \neg G \vee \neg K, \neg H \vee \neg K\}$

   1. $\neg G \vee \neg J$ [Premise]
   2. $\neg H \vee \neg J$ [Premise]
   3. $\neg G \vee \neg K$ [Premise]
   4. $\neg H \vee \neg K$ [Premise]
   5. G [Premise]
   6. J [¬ Conclusion]
   7. $\neg G$ [1, 6 Resolution]
   8. _ [5, 7 Resolution]

   Hence KB entails $\neg J$

2. KB= $\{P \rightarrow \neg Q, \neg Q \rightarrow R\}$. Show that KB $\vdash P \rightarrow R$

   Solution:

   1. $\neg P \vee \neg Q$ [Premise]
   2. $Q \vee R$ [Premise]
   3. P [¬ Conclusion]

4. $\neg R$ [$\neg$ Conclusion]
5. $\neg Q$ [1, 3 Resolution]
6. R [2, 5 Resolution]
7. _ [4, 6 Resolution]

Hence, KB $\vdash P \rightarrow R$

3. $\vdash ((P \vee Q) \wedge \neg P) \rightarrow Q$

Clausal form of $\neg(((P \vee Q) \wedge \neg P) \rightarrow Q)$ is $\{P \vee Q, \neg P, \neg Q\}$

1. $P \vee Q$ [$\neg$ Conclusion]
2. $\neg P$ [$\neg$ Conclusion]
3. $\neg Q$ [$\neg$ Conclusion]
4. Q [1, 2 Resolution]
5. _ [3, 4 Resolution]

Forward and backward chaining

The completeness of resolution makes it a very important inference model. But in many practical situations full power of resolution is not needed. Real-world knowledge bases often contain only clauses of restricted kind called **Horn Clause.** A Horn clauses is disjunction of literals with at most one positive literal

Three important properties of Horn clause are:

- ✓ Can be written as an implication
- ✓ Inference through forward chaining and backward chaining.
- ✓ Deciding entailment can be done in a time linear size of the knowledge base.

Forward chaining:

Idea: fire any rule whose premises are satisfied in the *KB*,

- add its conclusion to the *KB*, until query is found

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

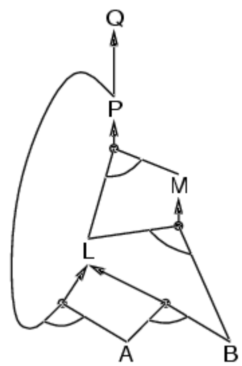
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

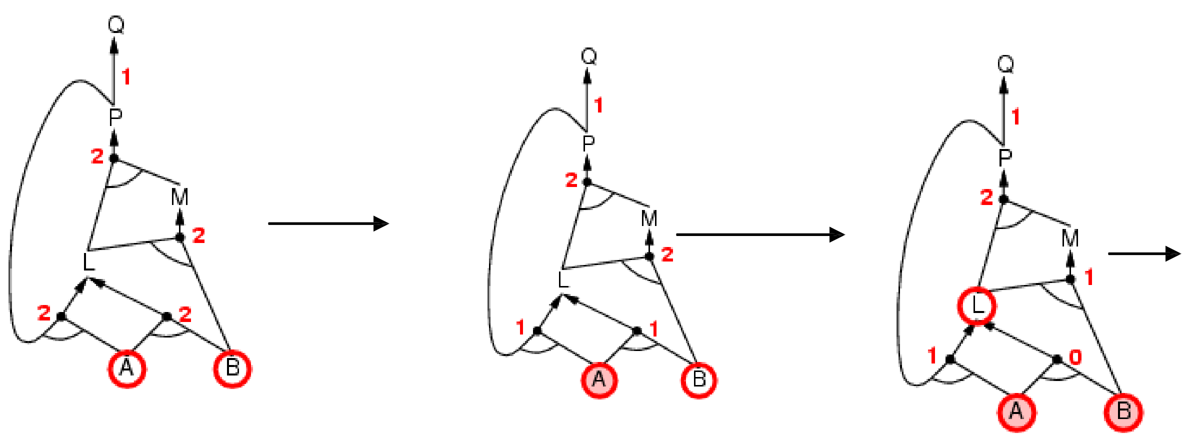
A

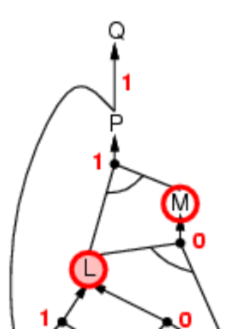
B

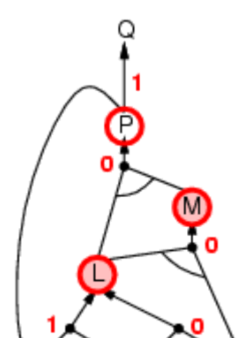


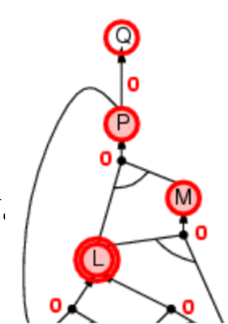
Prove that Q can be inferred from above KB

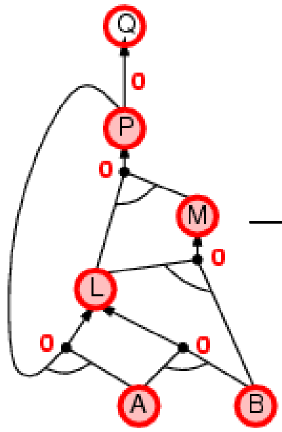
Solution:

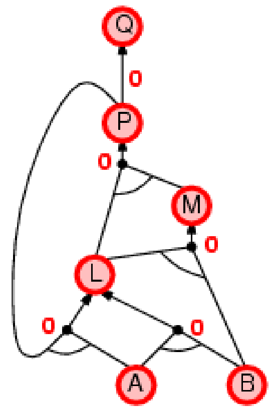












<u>Backward chaining:</u>

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*

For example, for above KB (as in forward chaining above)

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

Prove that Q can be inferred from above KB

<u>Solution:</u>
We know $P \Rightarrow Q$, try to prove P
$L \wedge M \Rightarrow P$
Try to prove L and M
$B \wedge L \Rightarrow M$
$A \wedge P \Rightarrow L$
Try to prove B, L and A and P
A and B is already known, since $A \wedge B \Rightarrow L$, L is also known
Since, $B \wedge L \Rightarrow M$, M is also known
Since, $L \wedge M \Rightarrow P$, p is known, hence the **proved.**

First-Order Logic

Pros and cons of propositional logic

- Propositional logic is declarative
- Propositional logic allows partial/disjunctive/negated information
    - (unlike most data structures and databases)
- Propositional logic is compositional:
    - meaning of $B \wedge P$ is derived from meaning of B and of P
- Meaning in propositional logic is context-independent
    - (unlike natural language, where meaning depends on context)
- Propositional logic has very limited expressive power
    - (unlike natural language)

Propositional logic assumes the world contains facts, whereas first-order logic (like natural language) assumes the world contains:

- Objects: people, houses, numbers, colors, baseball games, wars, …
- Relations: red, round, prime, brother of, bigger than, part of, comes between,…
- Functions: father of, best friend, one more than, plus, …

Logics in General

The primary difference between PL and FOPL is their ontological commitment:

Ontological Commitment: What exists in the world — TRUTH

- PL: facts hold or do not hold.
- FL : objects with relations between them that hold or do not hold

Another difference is:

Epistemological Commitment: What an agent believes about facts — BELIEF

Language	Ontological Commitment	Epistemological Commitment
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief $\in [0,1]$
Fuzzy logic	degree of truth $\in [0,1]$	known interval value

FOPL: Syntax

Predicate Logic: Syntax

Sentence	→	*AtomicSentence*
	|	*(Sentence* Connective *Sentence)*
	|	Quantifier *Variable, ... Sentence*
	|	¬ *Sentence*
AtomicSentence	→	*Predicate(Term, ...)* | *Term = Term*
Term	→	*Function(Term, ...)* | *Constant* | *Variable*
Connective	→	∧ | ∨ | ⇒ | ⇔
Quantifier	→	∀ | ∃
Constant	→	*A, B, C,* X_1, X_2, *Jim, Jack*
Variable	→	*a, b, c,* x_1, x_2, *counter, position, ...*
Predicate	→	*Adjacent-To, Younger-Than, HasColor, ...*
Function	→	*Father-Of, Square-Position, Sqrt, Cosine*

ambiguities are resolved through precedence or parentheses

Representing knowledge in first-order logic

The objects from the real world are represented by constant symbols (a,b,c,...). For instance, the symbol –Tom‖ may represent a certain individual called Tom.

Properties of objects may be represented by predicates applied to those objects (P(a), ...): e.g "male(Tom)" represents that Tom is a male.

Relationships between objects are represented by predicates with more arguments: "father(Tom, Bob)" represents the fact that Tom is the father of Bob.

The value of a predicate is one of the boolean constants T (i.e. true) or F (i.e. false)."father(Tom, Bob) = T" means that the sentence "Tom is the father of Bob" is true. "father(Tom, Bob) = F" means that the sentence "Tom is the father of Bob" is false.

Besides constants, the arguments of the predicates may be functions (f,g,...) or variables (x,y,...).

Function symbols denote mappings from elements of a domain (or tuples of elements of domains) to elements of a domain. For instance, weight is a function that maps objects to their weight: weight (Tom) = 150.Therefore the predicate greater-than (weight (Bob), 100) means that the weight of Bob is greater than 100. The arguments of a function may themselves be functions.

Variable symbols represent potentially any element of a domain and allow the formulation of general statements about the elements of the domain.

The quantifier‘s ∀ and ∃ are used to build new formulas from old ones.
"∃x P(x)" expresses that there is at least one element of the domain that makes P(x) true.
"∃x mother(x, Bob)‖ means that there is x such that x is mother of Bob or, otherwise stated, Bob has a mother.
"∀x P(x)" expresses that for all elements of the domain P(x) is true.

Quantifiers

Allows us to express properties of collections of objects instead of enumerating objects by name. Two quantifiers are:

Universal: –for all‖ ∀

Existential: –there exists‖ ∃

Universal quantification:

∀<*Variables*> <*sentence*>

Eg: Everyone at UAB is smart:

∀x At(x,UAB) ⇒ Smart(x)

∀x *P* is true in a model *m* iff *P* is true for all *x* in the model

Roughly speaking, equivalent to the conjunction of instantiations of *P*

At(KingJohn,UAB) ⇒ Smart(KingJohn) ∧ At(Richard,UAB) ⇒ Smart(Richard)∧At(UAB,UAB) ⇒ Smart(UAB)∧ ...

Typically, ⇒ is the main connective with ∀

- A universally quantifier is also equivalent to a set of implications over all objects

Common mistake: using ∧ as the main connective with ∀:

∀x At(x, UAB) ∧ Smart(x)

Means –Everyone is at UAB and everyone is smart‖

Existential quantification

∃<*variables*> <*sentence*>
Someone at UAB is smart:

$\exists x$ At(x, UAB) $\wedge$ Smart(x)

$\exists x$ *P* is true in a model *m* iff *P* is true for at least one *x* in the model

Roughly speaking, equivalent to the disjunction of instantiations of *P*

At(KingJohn,UAB) $\wedge$ Smart(KingJohn)$\vee$At(Richard,UAB) $\wedge$ Smart(Richard)
$\vee$At(UAB, UAB) $\wedge$ Smart(UAB) $\vee$ ...

Typically, $\wedge$ is the main connective with $\exists$

Common mistake: using $\Rightarrow$ as the main connective with $\exists$:
$\exists x$ At(x, UAB) $\Rightarrow$ Smart(x) is true even if there is anyone who is not at UAB!

FOPL: Semantic

An interpretation is required to give semantics to first-order logic. The interpretation is a non-empty –domain of discourse‖ (set of objects). The truth of any formula depends on the interpretation.

The interpretation provides, for each:
constant symbol an object in the domain
function symbols a function from domain tuples to the domain
predicate symbol a relation over the domain (a set of tuples)

Then we define:

universal quantifier $\forall x P(x)$ is True iff $P(a)$ is True for all assignments of domain elements *a* to *x*

existential quantifier $\exists x P(x)$ is True iff $P(a)$ is True for at least one assignment of domain element *a* to *x*

FOPL: Inference (Inference in first-order logic)

First order inference can be done by converting the knowledge base to PL and using propositional inference.

- How to convert universal quantifiers?
  - Replace variable by ground term.
- How to convert existential quantifiers?
  - Skolemization.

Universal instantiation (UI)

Substitute ground term (term without variables) for the variables.

For example consider the following KB

$\forall$ x King (x) $\wedge$ Greedy (x) $\Rightarrow$ Evil(x)
King (John)
Greedy (John)
Brother (Richard, John)

It's UI is:

King (John) $\wedge$ Greedy (John) $\Rightarrow$ Evil(John)
King (Richard) $\wedge$ Greedy (Richard) $\Rightarrow$ Evil(Richard)
King (John)
Greedy (John)
Brother (Richard, John)

Note: Remove universally quantified sentences after universal instantiation.

Existential instantiation (EI)

For any sentence α and variable v in that, introduce a constant that is not in the KB (called skolem constant) and substitute that constant for v.

E.g.: Consider the sentence, $\exists$ x Crown(x) $\wedge$ OnHead(x, John)

After EI,

Crown(C1) $\wedge$ OnHead(C1, John) where C1 is Skolem Constant.

Towards Resolution for FOPL:

- Based on resolution for propositional logic
- Extended syntax: allow variables and quantifiers
- Define –clausal form‖ for first-order logic formulae (CNF)
- Eliminate quantifiers from clausal forms
- Adapt resolution procedure to cope with variables (unification)

Conversion to CNF:

1. Eliminate implications and bi-implications as in propositional case
2. Move negations inward using De Morgan's laws

   plus rewriting $\neg\forall xP$ as $\exists x\neg P$ and $\neg\exists xP$ as $\forall x\neg P$
3. Eliminate double negations
4. Rename bound variables if necessary so each only occurs once

   e.g. $\forall xP(x)\vee\exists xQ(x)$ becomes $\forall xP(x)\vee\exists yQ(y)$
5. Use equivalences to move quantifiers to the left

   e.g. $\forall xP(x)\wedge Q$ becomes $\forall x\,(P(x)\wedge Q)$ where x is not in Q

   e.g. $\forall xP(x)\wedge\exists yQ(y)$ becomes $\forall x\exists y(P(x)\wedge Q(y))$
6. Skolemise (replace each existentially quantified variable by a **new** term)

   $\exists xP(x)$ becomes $P(a_0)$ using a Skolem constant a_0 since $\exists x$ occurs at the outermost level

$\forall x \exists y P(x, y)$ becomes $P(x, f_0(x))$ using a Skolem function f_0 since $\exists y$ occurs within $\forall x$

7. The formula now has only universal quantifiers and all are at the left of the formula: drop them
8. Use distribution laws to get CNF and then clausal form

Example:

1.) $\forall x\ [\forall y P(x, y) \rightarrow \neg \forall y (Q(x, y) \rightarrow R(x, y))]$

Solution:

1. $\forall x\ [\neg \forall y P(x, y) \vee \neg \forall y (\neg Q(x, y) \vee R(x, y))]$

2, 3. $\forall x\ [\exists y \neg P(x, y) \vee \exists y (Q(x, y) \wedge \neg R(x, y))]$

4. $\forall x\ [\exists y \neg P(x, y) \vee \exists z\ (Q(x, z) \wedge \neg R(x, z))]$

5. $\forall x \exists y \exists z\ [\neg P(x, y) \vee (Q(x, z) \wedge \neg R(x, z))]$

6. $\forall x\ [\neg P(x, f(x)) \vee (Q(x, g(x)) \wedge \neg R(x, g(x)))]$

7. $\neg P(x, f(x)) \vee (Q(x, g(x)) \wedge \neg R(x, g(x)))$

8. $(\neg P(x, f(x)) \vee Q(x, g(x))) \wedge (\neg P(x, f(x)) \vee \neg R(x, g(x)))$

8. $\{\neg P(x, f(x)) \vee Q(x, g(x)),\ \neg P(x, f(x)) \vee \neg R(x, g(x))\}$

2.) $\neg \exists x \forall y \forall z\ ((P(y) \vee Q(z)) \rightarrow (P(x) \vee Q(x)))$

Solution:

1. $\neg \exists x \forall y \forall z\ (\neg (P(y) \vee Q(z)) \vee P(x) \vee Q(x))$

2. $\forall x \neg \forall y \forall z\ (\neg (P(y) \vee Q(z)) \vee P(x) \vee Q(x))$

2. $\forall x \exists y \neg \forall z\ (\neg (P(y) \vee Q(z)) \vee P(x) \vee Q(x))$

2. $\forall x \exists y \exists z \neg (\neg (P(y) \vee Q(z)) \vee P(x) \vee Q(x))$

2. $\forall x \exists y \exists z\ ((P(y) \vee Q(z)) \wedge \neg (P(x) \vee Q(x)))$

6. $\forall x\ ((P(f(x)) \vee Q(g(x))) \wedge \neg P(x) \wedge \neg Q(x))$

7. $(P(f(x)) \vee Q(g(x)) \wedge \neg P(x) \wedge \neg Q(x)$

8. $\{P(f(x)) \vee Q(g(x)),\ \neg P(x),\ \neg Q(x)\}$

Unification:

A unifier of two atomic formulae is a substitution of terms **for variables** that makes them identical.

- Each variable has at most one associated term
- Substitutions are applied simultaneously

Unifier of $P(x, f(a), z)$ and $P(z, z, u)$: $\{x/ f(a), z/ f(a), u/ f(a)\}$

We can get the inference immediately if we can find a substitution α such that *King(x)* and *Greedy(x)* match *King(John)* and *Greedy(y)*

α = {x/John,y/John} works

Unify(α ,β) = θ if αθ = θβ

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}
Knows(John,x)	Knows(y,OJ)	{x/OJ,y/John}
Knows(John,x)	Knows(y,Mother(y))	{y/John,x/Mother(John)}}
Knows(John,x)	Knows(x,OJ)	{fail}

Last unification is failed due to overlap of variables. x can not take the values of John and OJ at the same time.

We can avoid this problem by renaming to avoid the name clashes (standardizing apart)

E.g.

Unify{Knows(John,x) Knows(z,OJ) } = {x/OJ, z/John}

Let C1 and C2 be two clauses. If C1 and C2 have no variables in common, then they are said to be standardized apart. Standardized apart eliminates overlap of variables to avoid clashes by renaming variables.

Another complication:

To unify *Knows(John,x)* and *Knows(y,z)*,
Unification of Knows*(John,x)* and *Knows(y,z)* gives α ={y/John, x/z } or α={y/John, x/John, z/John}

First unifier gives the result Knows(John,z) and second unifier gives the resultKnows(John, John). Second can be achieved from first by substituting john in place of z. The first unifier is more general than the second.

There is a single most general unifier (MGU) that is unique up to renaming of variables.

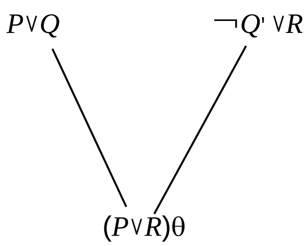
MGU = { y/John, x/z }

Towards Resolution for First-Order Logic

- Based on resolution for propositional logic
- Extended syntax: allow variables and quantifiers
- Define –clausal form‖ for first-order logic formulae
- Eliminate quantifiers from clausal forms
- Adapt resolution procedure to cope with variables (unification)

First-Order Resolution

For clauses $P \lor Q$ and $\neg Q' \lor R$ with Q, Q' atomic formulae



where θ is a most general unifier for Q and Q'

$(P \lor R)\theta$ is the resolvent of the two clauses

Applying Resolution Refutation

- Negate query to be proven (resolution is a refutation system)
- Convert knowledge base and negated query into CNF and extract clauses
- Repeatedly apply resolution to clauses or copies of clauses until either the empty clause (contradiction) is derived or no more clauses can be derived (a copy of a clause is the clause with all variables renamed)
- If the empty clause is derived, answer ‗yes‘ (query follows from knowledge base), otherwise answer ‗no‘ (query does not follow from knowledge base)

Resolution: Examples

1.) $\vdash \exists x\, (P(x) \rightarrow \forall x P(x))$

Solution:

Add negation of the conclusion and convert the predicate in to CNF:

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

1, 2. $\forall x \neg (\neg P(x) \lor \forall x P(x))$

2. $\forall x\, (\neg\neg P(x) \land \neg \forall x P(x))$

2, 3. $\forall x\, (P(x) \land \exists x \neg P(x))$

4. $\forall x\,(P(x) \wedge \exists y\, \neg P(y))$

5. $\forall x \exists y (P(x) \wedge \neg P(y))$

6. $\forall x\,(P(x) \wedge \neg P(f(x)))$

8. $P(x), \neg P(f(x))$

Now, we can use resolution as;

1. $P(x)$ [$\neg$ Conclusion]

2. $\neg P(f(y))$ [Copy of $\neg$ Conclusion]

3. _ [1, 2 Resolution $\{x/f(y)\}$]

2.) $\vdash \exists x \forall y \forall z\, ((P(y) \vee Q(z)) \rightarrow (P(x) \vee Q(x)))$

<u>Solution:</u>

1. $P(f(x)) \vee Q(g(x))$ [$\neg$ Conclusion]

2. $\neg P(x)$ [$\neg$ Conclusion]

3. $\neg Q(x)$ [$\neg$ Conclusion]

4. $\neg P(y)$ [Copy of 2]

5. $Q(g(x))$ [1, 4 Resolution $\{y/f(x)\}$]

6. $\neg Q(z)$ [Copy of 3]

7. _ [5, 6 Resolution $\{z/g(x)\}$]

3.)

The following axioms describe the situation:

1. If the coin comes up heads, then I win.
2. If it comes up tails, then you lose.
3. If it does not come up heads, then it comes up tails.
4. if you lose, then I win.

Which may be represented as:

1. $H \rightarrow W(me)$ //H: heads , W: win
2. $T \rightarrow L(you)$ //T: tails, L: lose
3. $\neg H \rightarrow T$
4. $L(you) \rightarrow W(me)$

Next, our argument is converted to clause form

1. $\neg H \vee W(me)$
2. $\neg T \vee L(you)$
3. $H \vee T$
4. $\neg L(you) \vee W(me)$

Then, add the negation of the conclusion

5. $\neg W(me)$ //also in clause form

Finally, we attempt to obtain a contradiction

2,4	$\neg T \vee W(me)$	6
1,3	$T \vee W(me)$	7
6,7	$W(me)$	8
5,8	□	**//contradiction!**

Hence W(me) //I win!!

Q.) Anyone passing his history exams and winning the lottery is happy. But anyone who studies or is lucky can pass all his exams. John did not study but John is lucky. Anyone who is lucky wins the lottery. Is John happy?

1. Anyone passing his history exams and winning the lottery is happy.

   $\forall x\ Pass(x, History) \wedge Win(x, Lottery) \Rightarrow Happy(x)$

2. But anyone who studies or is lucky can pass all his exams.

   $\forall x\ \forall y\ Study(x) \vee Lucky(x) \Rightarrow Pass(x,y)$

3. John did not study, but John is lucky

   $\neg\ Study(John) \wedge Lucky(John)$

4. Anyone who is lucky wins the lottery.

   $\forall x\ Lucky(x) \Rightarrow Win(x, Lottery)$

Now, Convert the KB to CNF:

Eliminate implications:

1. $\forall x\ \neg\ (Pass(x, History) \wedge Win(x, Lottery)) \vee Happy(x)$
2. $\forall x\ \forall y\ \neg\ (Study(x) \vee Lucky(x)\ ) \vee Pass(x,y)$
3. $\neg\ Study(John) \wedge Lucky(John)$
4. $\forall x\ \neg\ Lucky(x) \vee Win(x, Lottery)$

Move $\neg$ inward

1. $\forall x\ \neg\ Pass(x, History) \vee \neg\ Win(x, Lottery)) \vee Happy(x)$
2. $\forall x\ \forall y\ (\neg\ Study(x) \wedge \neg Lucky(x)\ ) \vee Pass(x,y)$
3. $\neg\ Study(John) \wedge Lucky(John)$
4. $\forall x\ \neg\ Lucky(x) \vee Win(x, Lottery)$

Distribute $\wedge$ over $\vee$

1. $\neg$ Pass(x, History) $\vee$ $\neg$ Win(x, Lottery)) $\vee$ Happy(x)

2. ($\neg$ Study(x) $\vee$ Pass(x,y)) $\wedge$ ($\neg$ Lucky(x) $\vee$ Pass(x,y))

3. $\neg$ Study(John) $\wedge$ Lucky(John)

4. $\neg$ Lucky(x) $\vee$ Win(x, Lottery)

Now, the KB contains:

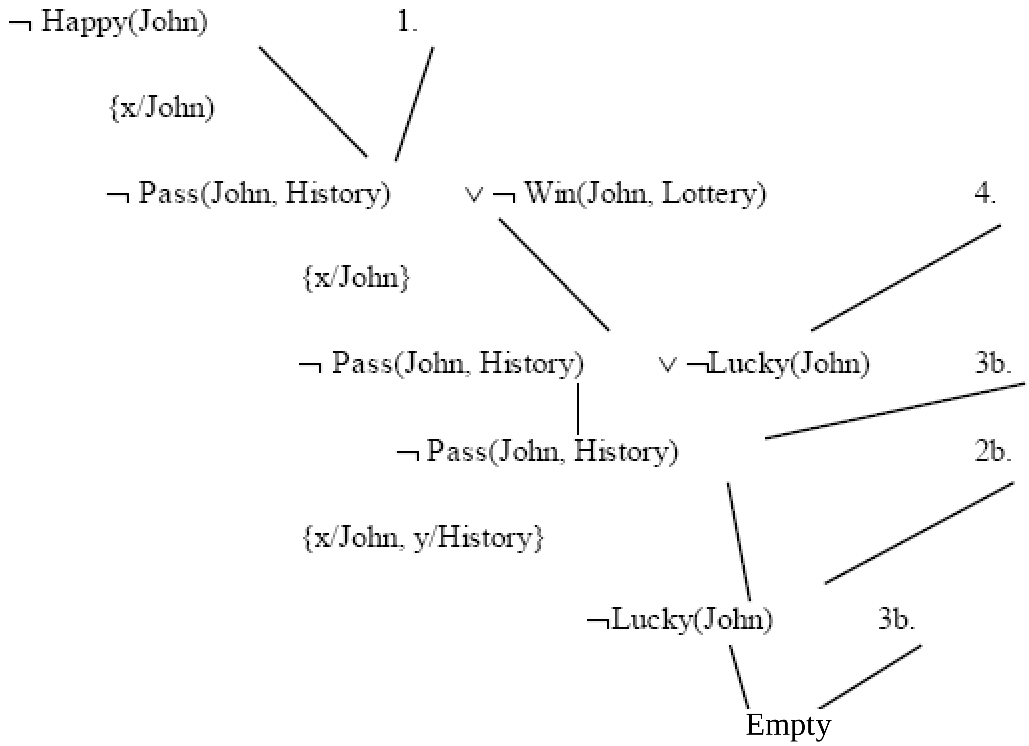
1. $\neg$ Pass(x, History) $\vee$ $\neg$ Win(x, Lottery) $\vee$ Happy(x)
2. a. $\neg$ Study(x) $\vee$ Pass(x,y)
2. b. $\neg$Lucky(x) $\vee$ Pass(x,y)
3. a. $\neg$ Study(John)
   b. Lucky(John)
4. $\neg$ Lucky(x) $\vee$ Win(x, Lottery)

Standardize the variables apart:

1. $\neg$ Pass(x1, History) $\vee$ $\neg$ Win(x1, Lottery) $\vee$ Happy(x1)
2. a. $\neg$ Study(x2) $\vee$ Pass(x2,y1)
2. b. $\neg$Lucky(x3) $\vee$ Pass(x3,y2)
3. a. $\neg$ Study(John)
   b. Lucky(John)
4. $\neg$ Lucky(x4) $\vee$ Win(x4, Lottery)

5. $\neg$ Happy(John) **(Negation of the conclusion added)**

<u>Now Use resolution as below:</u>


¬ Happy(John)
1.
{x/John)
¬ Pass(John, History) ∨ ¬ Win(John, Lottery)
4.
{x/John}
¬ Pass(John, History) ∨ ¬Lucky(John)
3b.
¬ Pass(John, History)
2b.
{x/John, y/History}
¬Lucky(John)
3b.
Empty

Symbolic versus statistical reasoning:

The (Symbolic) methods basically represent uncertainty belief as being

- True,
- False, *or*
- Neither True nor False.

Some methods also had problems with

- Incomplete Knowledge
- Contradictions in the knowledge.

Statistical methods provide a method for representing beliefs that are not certain (or uncertain) but for which there may be some supporting (or contradictory) evidence.

Statistical methods offer advantages in two broad scenarios:

Genuine Randomness

-- Card games are a good example. We may not be able to predict any outcomes with certainty but we have knowledge about the likelihood of certain items (*e.g.* like being dealt an ace) and we can exploit this.

Exceptions

-- Symbolic methods can represent this. However if the number of exceptions is large such system tend to break down. Many common sense and expert reasoning tasks for example. Statistical techniques can *summarise* large exceptions without resorting enumeration.

Basic Statistical methods – Probability:

The basic approach statistical methods adopt to deal with uncertainty is via the axioms of probability:

- Probabilities are (real) numbers in the range 0 to 1.
- A probability of $P(A) = 0$ indicates total uncertainty in A, $P(A) = 1$ total certainty and values in between some degree of (un)certainty.
- Probabilities can be calculated in a number of ways.

  Very Simply

  Probability = (number of desired outcomes) / (total number of outcomes)

  So given a pack of playing cards the probability of being dealt an ace from a full normal deck is 4 (the number of aces) / 52 (number of cards in deck) which is 1/13. Similarly the probability of being dealt a spade suit is 13 / 52 = 1/4.

Conditional probability, $P(A|B)$, indicates the probability of of event A given that we know event B has occurred.

The aim of a **probabilistic logic** (or **probability logic**) is to combine the capacity of probability theory to handle uncertainty with the capacity of deductive logic to exploit structure. The result is a richer and more expressive formalism with a broad range of possible application areas. Probabilistic logic is a natural extension of traditional logic truth tables: the results they define are derived through probabilistic expressions instead. The difficulty with probabilistic logics is that they tend to multiply the computational complexities of their probabilistic and logical components.

Random Variables:

In probability theory and statistics, a **random variable** (or **stochastic variable**) is a way of assigning a value (often a real number) to each possible outcome of a random event. These values might represent the possible outcomes of an experiment, or the potential values of a quantity whose value is uncertain (e.g., as a result of incomplete information or imprecise measurements.) Intuitively, a random variable can be thought of as a quantity whose value is not fixed, but which can take on different values; normally, a probability distribution is used to describe the probability of different values occurring. Random variables are usually real-valued, but one can consider arbitrary types such as boolean values, complex numbers, vectors, matrices, sequences, trees, sets, shapes, manifolds and functions. The term *random element* is used to encompass all such related concepts.

For example: There are two possible outcomes for a coin toss: heads, or tails. The possible outcomes for one fair coin toss can be described using the following random variable:

$$X = \begin{cases} \text{head}, \\ \text{tail}. \end{cases}$$

and if the coin is equally likely to land on either side then it has a probability mass function given by:

$$\rho_X(x) = \begin{cases} \frac{1}{2}, & \text{if } x = \text{head}, \\ \frac{1}{2}, & \text{if } x = \text{tail}. \end{cases}$$

Example: A simple world consisting of two random variables:
Cavity– a Boolean variable that refers to whether my lower left wisdom tooth has a cavity
Toothache- a Boolean variable that refers to whether I have a toothache or not.

We use the single capital letters to represent unknown random variables
P induces a probability distribution for any random variables X.

Each RV has a domain of values that it can take it, e. g. domain of *Cavity* is {true, false}

RVs domain are: Boolean, Discrete and Continuous

Atomic Event:

An **atomic event**is a complete specification of the state of the world about which the agent is uncertain.

Example:
In the above world with two random variables (Cavity and Toothache) there are only four distinct atomic events, one being:

Cavity = false, Toothache = true

Which are the other three atomic events?

Propositions:

Think of a proposition as the event (set of sample points) where the proposition is true

Given Boolean random variables A and B:

event α = set of sample points where A(ω) = true event
¬α = set of sample points where A(ω) = false event a ^
b = points where A(ω) = true and B(ω) = true

Often in AI applications, the sample points are defined by the values of a set of random variables, i.e., the sample space is the Cartesian product of the ranges of the variables.

With Boolean variables, sample point = propositional logic model

e.g., A = true, B = false, or a ^ ¬b.

Proposition = disjunction of atomic events in which it is true

e.g., (a ∨ b) ≡ (¬a ^ b) ∨(a ^ ¬b) ∨(a ^ b)

P(a ∨ b) = P(¬a^ b) + P(a^ ¬b) + P(a^ b)

Propositional or Boolean random variables

e.g., *Cavity*(do I have a cavity?)

Discrete random variables (finite or infinite)

e.g., *Weather* is one of *(sunny, rain, cloudy, snow)*

Weather = rain is a proposition

Values must be exhaustive and mutually exclusive
Continuous random variables (bounded or unbounded)
e.g., *Temp = 21.6*, also allow, e.g., *Temp < 22.0*.

Prior Probability:

The prior or unconditional probability associated with a proposition is the degree of belief accorded to it in the absence of any other information.

Example:
P(Weather= sunny) = 0.72, P(Weather= rain) = 0.1, P(Weather= cloudy) = 0.08, P(Weather= snow) = 0.1

Probability distribution gives values for all possible assignments:

P(Weather) = (0.72, 0.1, 0.08, 0.1)

Joint probability distribution for a set of r.v.s gives the probability of every atomic event on those r.v.s (i.e., every sample point)

P(Weather, Cavity) = a 4 ×2 matrix of values.

$Weather =$	$sunny$	$rain$	$cloudy$	$snow$
$Cavity = true$	0.144	0.02	0.016	0.02
$Cavity = false$	0.576	0.08	0.064	0.08

Every question about a domain can be answered by the joint distribution because every event is a sum of sample points

Conditional Probability:

The conditional probability –P(a|b)‖ is the probability of –a‖ given that all we know is –b‖.

Example: *P(cavity|toothache) = 0.8* means if a patient have toothache and no other information is yet available, then the probability of patient‘s having the cavity is 0.8.

Definition of conditional probability:

$$P(a|b) = P(a \wedge b)/P(b) \text{ if } P(b) \neq 0$$

Product rule gives an alternative formulation:

$$P(a \wedge b) = P(a|b)P(b) = p(b|a)P(a)$$

Inference using full joint probability distribution:

We use the full joint distribution as the knowledge base from which answers to all questions may be derived. The probability of a proposition is equal to the sum of the probabilities of the atomic events in which it holds.

$$P(a) = \Sigma P(ei)$$

Therefore, given a full joint distribution that specifies the probabilities of all the atomic events, one can compute the probability of any proposition.

Full Joint probability distribution : an example

We consider the following domain consisting of three Boolean variables: Toothache, Cavity, and Catch (the dentist's nasty steel probe catches in my tooth).

The full joint distribution is the following 2x2x2 table:

	toothache		¬toothache	
	catch	¬catch	catch	¬catch
cavity	0.108	0.012	0.072	0.008
¬cavity	0.016	0.064	0.144	0.576

The probability of any proposition can be computed from the probabilities in the table. The probabilities in the joint distribution must sum to 1.

Each cell represents an atomic event and these are all the possible atomic events.

P(cavity or toothache) =

P(cavity, toothache, catch) + P(cavity, toothache, ¬catch) + P(cavity, ¬toothache, catch) + P(cavity, ¬toothache, ¬catch) + P(¬cavity, toothache, catch) + P(¬cavity, toothache, ¬catch)

= 0.108+0.012+0.072+0.008+0.016+0.064=0.28

We simply identify those atomic events in which the proposition is true and add up their probabilities

Bayes' Rule (Theorem) :

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

Why is the Bayes' rule is useful in practice? Bayes' rule is useful in practice because there are many cases where we have good probability estimates for three of the four probabilities involved, and therefore can compute the fourth one.

Useful for assessing diagnostic probability from causal probability:

$$P(Cause|Effect) = \frac{P(Effect|Cause)P(Cause)}{P(Effect)}$$

Diagnostic knowledge is often more fragile than causal knowledge.

Example of Bayes' rule:

A doctor knows that the disease meningitis causes the patient tohave a stiff neck 50% of the time. The doctor also knows that the probability that a patient has meningitis is 1/50,000, and the probability that any patient has a stiff neck is 1/20.

Find the probability that a patient with a stiff neck has meningitis.

Here, we are given;

$p(s|m) = 0.5$
$p(m) = 1/50000$
$p(s) = 1/20$

Now using Bayes‘ rule;

$P(m|s) = P(s|m)P(m)/P(s) = (0.5*1/50000)/(1/20) = 0.0002$

Uses of Bayes' Theorem :

In doing an expert task, such as medical diagnosis, the goal is to determine identifications (diseases) given observations (symptoms). Bayes' Theorem provides such a relationship.

$P(A \mid B) = P(B \mid A) * P(A) / P(B)$

Suppose: *A* = Patient has measles, *B* = has a rash

Then: *P(measles/rash) =* *P(rash/measles) * P(measles) / P(rash)*

The desired diagnostic relationship on the left can be calculated based on the known statistical quantities on the right.

Bayesian networks:

- *A data structure to represent the dependencies among variables and to give a concise specification of any full joint probability distribution.*
- Also called *belief networks* or *probabilistic network* or *casual network* or *knowledge map*.

The basic idea is:

- Knowledge in the world is *modular* -- most events are conditionally independent of most other events.
- Adopt a model that can use a more local representation to allow interactions between events that *only* affect each other.
- Some events may only be *unidirectional* others may be *bidirectional* -- make a distinction between these in model.
- Events may be causal and thus get chained together in a network.

A Bayesian network is a directed acyclic graph which consists of:

- A set of random variables which makes up the nodes of the network.
- A set of directed links (arrows) connecting pairs of nodes. If there is an arrow from node X to node Y, X is said to be a parent of Y.
- Each node Xi has a conditional probability distribution P(Xi| Parents(Xi)) that quantifies the effect of the parents on the node.

Intuitions:

- A Bayesian network models our incomplete understanding of the causal relationships from an application domain.
- A node represents some state of affairs or event.
- **A link from X to Y means that X has a direct influence on Y.**

Implementation

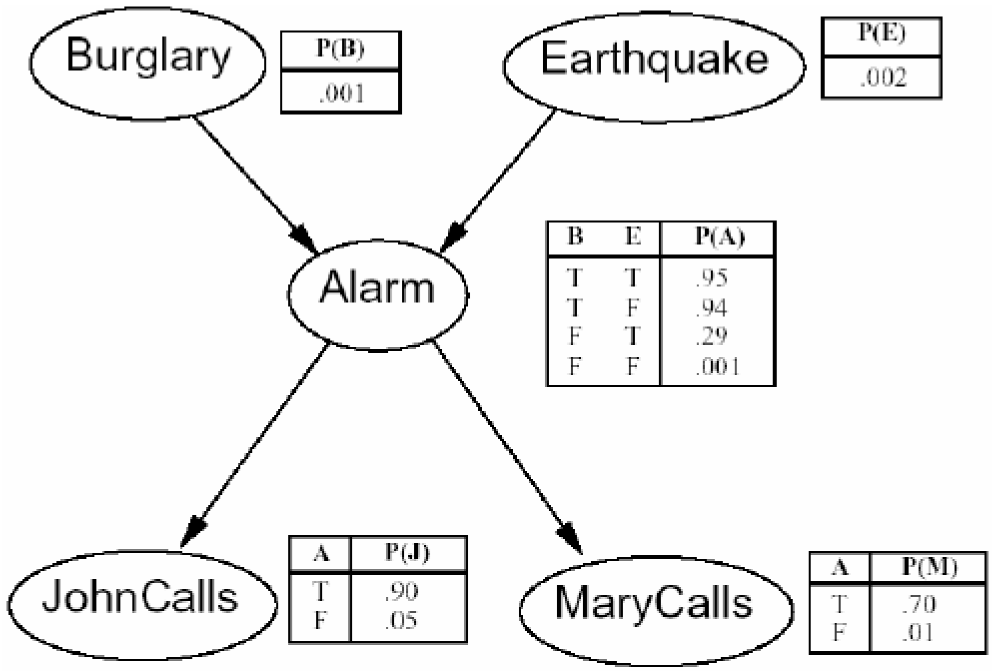
- A *Bayesian Network* is a *directed acyclic graph*:
  - A graph where the directions are links which indicate dependencies that exist between nodes.
  - Nodes represent propositions about events or events themselves.
  - Conditional probabilities quantify the strength of dependencies.

Example:

Sample Domain:

You have a burglar alarm installed in your home. It is fairly reliable at detecting a burglary, but also responds on occasion to minor earthquakes. You also have two neighbors, John and Mary, who have promised to call you at work when they hear the alarm. John always calls when he hears the alarm, but sometimes confuses the telephone ringing with the alarm and calls then, too. Mary, on the other hand, likes rather loud music and sometimes misses the alarm altogether.
We would like have to estimate the probability of a burglary with given evidence who has or has not call.

Variables: Burglary, Earthquake, Alarm, JohnCalls, MaryCalls



The probabilities associated with the nodes reflect our representation of the causal relationships.

A Bayesian network provides a complete description of the domain in the sense that one can compute the probability of any state of the world (represented as a particular assignment to each variable).

Example: What is the probability that the alarm has sounded, but neitherburglary nor an earthquake has occurred, and both John and Mary call?

P(j, m, a, ¬b, ¬e) = P(j|a) P(m|a) P(a|, ¬b, ¬e) P(¬b) P(¬e)

= 0.90*0.70*0.001*0.999*0.998 = 0.00062

Consider the following example:

- The probability, $P(s_1)$ that my car won't start.
- If my car won't start then it is likely that
  - The battery is flat or
  - The staring motor is broken.

In order to decide whether to fix the car myself or send it to the garage I make the following decision:

- If the headlights do not work then the battery is likely to be flat so i fix it myself.

- If the starting motor is defective then send car to garage.
- If battery and starting motor both gone send car to garage.

The network to represent this is as follows:

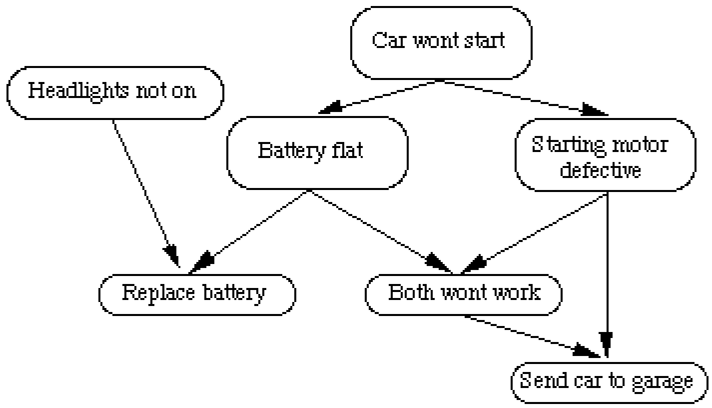


Fig. A simple Bayesian network

Reasoning in Bayesian nets:

(To be added more.........)

Keep visiting csitauthority.blogspot.com for updated content on this note from Mr. Bal Krishna Subedi