UNIT-03

Probabilistic Reasoning

UNIT-03/LECTURE-01

Bayes Theorem

This reads that given some evidence E then probability that hypothesis is true is equal to the ratio of the probability that E will be true given times the a priori evidence on the probability of and the sum of the probability of E over the set of all hypotheses times the probability of these hypotheses.

The set of all hypotheses must be mutually exclusive and exhaustive.
Thus to find if we examine medical evidence to diagnose an illness. We must know all the prior probabilities of find symptom and also the probability of having an illness based on certain symptoms being observed.

Bayesian statistics lie at the heart of most statistical reasoning systems.
How is Bayes theorem exploited?
The key is to formulate problem correctly:
P(A|B) states the probability of A given only B's evidence. If there is other relevant evidence then it must also be considered.

Herein lies a problem:
• All events must be mutually exclusive. However in real world problems events are not generally unrelated. For example in diagnosing measles, the symptoms of spots and a fever are related. This means that computing the conditional probabilities gets complex.
In general if a prior evidence, p and some new observation, N then computing
grows exponentially for large sets of p
• All events must be exhaustive. This means that in order to compute all probabilities the set of possible events must be closed. Thus if new information arises the set must be created afresh and all probabilities recalculated.

Thus Simple Bayes rule-based systems are not suitable for uncertain reasoning.
• Knowledge acquisition is very hard.
• Too many probabilities needed -- too large a storage space.
• Computation time is too large.
• Updating new information is difficult and time consuming.
• Exceptions like ``none of the above'' cannot be represented.
• Humans are not very good probability estimators.

However, Bayesian statistics still provide the core to reasoning in many uncertain reasoning systems with suitable enhancement to overcome the above problems.
We will look at three broad categories:
• Certainty factors,
• Dempster-Shafer models,
• Bayesian networks.

Belief Models and Certainty Factors
This approach has been suggested by Shortliffe and Buchanan and used in their famous medical diagnosis MYCIN system.
MYCIN is essentially and expert system. Here we only concentrate on the probabilistic reasoning aspects

of MYCIN.
• MYCIN represents knowledge as a set of rules.
• Associated with each rule is a certainty factor
• A certainty factor is based on measures of belief B and disbelief D of an hypothesis given evidence E as follows:
where is the standard probability.
• The certainty factor C of some hypothesis given evidence E is defined as:

Reasoning with Certainty factors
• Rules expressed as if evidence list then there is suggestive evidence with probability, p for symptom .
• MYCIN uses rules to reason backward to clinical data evidence from its goal of predicting a disease-causing organism.
• Certainty factors initially supplied by experts changed according to previous formulae.
• How do we perform reasoning when several rules are chained together?
Measures of belief and disbelief given several observations are calculated as follows:
• How about our belief about several hypotheses taken together? Measures of belief given several hypotheses and to be combined logically are calculated as follows:
Disbelief is calculated similarly.

Overcoming the Bayes Rule shortcomings
Certainty Factors do adhere to the rules of Bayesian statistics, but it can represent tractable knowledge systems:
• Individual rules contribute belief in an hypotheses -- basically a conditional probability.
• The formulae for combination of evidence / hypotheses basically assume that all rules are independent ruling out the need for joint probabilities.
• The burden of guaranteeing independence is placed on the rule writer.

UNIT-03/Lecture 02

Dempster-Shafer Models
This can be regarded as a more general approach to representing uncertainty than the Bayesian approach.
Bayesian methods are sometimes inappropriate:
Let A represent the proposition Demi Moore is attractive.Then the axioms of probability insist that
Now suppose that Andrew does not even know who Demi Moore is.
Then
• We cannot say that Andrew believes the proposition if he has no idea what it means.
• Also, It is not fair to say that he disbelieves the proposition.
• It would therefore be meaningful to denote Andrew's belief of B(A) and as both being 0.
• Certainty factors do not allow this.

Dempster-Shafer Calculus
The basic idea in representing uncertainty in this model is:
• Set up a confidence interval -- an interval of probabilities within which the true probability lies with a certain confidence -- based on the Belief B and plausibility PL provided by some evidence E for a proposition P.
• The belief brings together all the evidence that would lead us to believe in P with some certainty.
• The plausibility brings together the evidence that is compatible with P and is not inconsistent with it.
• This method allows for further additions to the set of knowledge and does not assume disjoint

outcomes.
If is the set of possible outcomes, then a mass probability, M, is defined for each member of the set and takes values in the range [0,1].
The Null set, , is also a member of .
NOTE: This deals wit set theory terminology that will be dealt with in a tutorial shortly. Also see exercises to get experience of problem solving in this important subject matter.
M is a probability density function defined not just for but for em all subsets.
So if is the set { Flu (F), Cold (C), Pneumonia (P) } then is the set { , {F}, {C}, {P}, {F, C}, {F, P}, {C, P}, {F, C, P} }
• The confidence interval is then defined as [B(E),PL(E)]
where i.e. all the evidence that makes us believe in the correctness of P, and

where i.e. all the evidence that contradicts P.
Combining beliefs
• We have the ability to assign M to a set of hypotheses.
• To combine multiple sources of evidence to a single (or multiple) hypothesis do the following:
o Suppose and are two belief functions.
o Let X be the set set of subsets of to which assigns a nonzero value and letY be a similar set for
o Then to get a new belief function from the combination of beliefs in and we do:
whenever .
NOTE: We define to be 0 so that the orthogonal sum remains a basic probability assignment.
Combining beliefs
• We have the ability to assign M to a set of hypotheses.
• To combine multiple sources of evidence to a single (or multiple) hypothesis do the following:
o Suppose and are two belief functions.
o Let X be the set of subsets of to which assigns a nonzero value and letY be a similar set for
o Then to get a new belief function from the combination of beliefs in and we do:
Whenever .
NOTE: We define to be 0 so that the orthogonal sum remains a basic probability assignment.

Bayesian networks also called Belief Networks or Probabilistic Inference Networks.
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.

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

Consider the following example:
• The probability, that my car won't start.
• If my car won't start then it is likely that
o The battery is flat or
o 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.

Reasoning in Bayesian nets
• Probabilities in links obey standard conditional probability axioms.
• Therefore follow links in reaching hypothesis and update beliefs accordingly.
• A few broad classes of algorithms have bee used to help with this:
o Pearls's message passing method.
o Clique triangulation.
o Stochastic methods.
o Basically they all take advantage of clusters in the network and use their limits on the influence to constrain the search through net.
o They also ensure that probabilities are updated correctly.
• Since information is local information can be readily added and deleted with minimum effect on the whole network. ONLY affected nodes need updating.

UNIT-03/Lecture 03

Fuzzy Logic
This topic is treated more formally in other courses. Here we summarize the main points for the sake completeness.
Fuzzy logic is a totally different approach to representing uncertainty:
• It focuses on ambiguities in describing events rather the uncertainty about the occurrence of an event.
• Changes the definitions of set theory and logic to allow this.
• Traditional set theory defines set memberships as a Boolean predicate.
Fuzzy Set Theory
• Fuzzy set theory defines set membership as a possibility distribution.

This basically states that we can take n possible events and us f to generate as single possible outcome. This extends set membership since we could have varying definitions of, say, hot curries. One person might declare that only curries of Vindaloo strength or above are hot whilst another might say madras and above are hot. We could allow for these variations definition by allowing both possibilities in fuzzy definitions.
Once set membership has been redefined we can develop new logics based on combining of
Uncertain Reasoning

Sometimes the knowledge in rules is not certain. Rules then may be enhanced by adding information about how certain the conclusions drawn from the rules may be. Here we describe certainty factors and their manipulation.
Often, experts can't give definite answers.
May require an inference mechanism that derives conclusions by combining uncertainties.

Fuzzy Inferencing
The process of fuzzy reasoning is incorporated into what is called a Fuzzy Inferencing System. It is comprised of three steps that process the system inputs to the appropriate system outputs. These steps are 1) Fuzzification, 2) Rule Evaluation, and 3) Defuzzification. The system is illustrated in the following figure. https://www.rgpvonline.com
1. Fuzzification : is the first step in the fuzzy inferencing process. This involves a domain formation where crisp inputs are transformed into fuzzy inputs. Crisp inputs are exact inputs measured by sensors and

passed into the control system for processing, such as temperature, pressure, rpm's, etc.. Each crisp input that is to be processed by the FIU has its own group of membership functions or sets to which they are transformed. This group of membership functions exists within a universe of discourse that holds all relevant values that the crisp input can possess. The following shows the structure of membership functions within a universe of discourse for a crisp input.

2. Degree of membership: degree to which a crisp value is compatible to a membership function, value from 0 to 1, also known as truth value or fuzzy input. membership function, MF: defines a fuzzy set by mapping crisp values from its domain to the sets associated degree of membership.

3.crisp inputs: distinct or exact inputs to a certain system variable, usually measured

4.parameters external from the control system, e.g. 6 Volts.

5.label: descriptive name used to identify a membership function.

6.scope: or domain, the width of the membership function, the range of concepts, usually numbers, over which a membership function is mapped.

7.universe of discourse: range of all possible values, or concepts, applicable to a system variable. When designing the number of membership functions for an input variable, labels must initially be determined for the membership functions. The number of labels correspond to the number of regions that the universe should be divided, such that each label describes a region of behavior. A scope must be assigned to each membership function that numerically identifies the range of input values that correspond to a label. The shape of the membership function should be representative of the variable. However this shape is also restricted by the computing resources available. Complicated shapes require more complex descriptive equations or large lookup tables. The next figure shows examples of possible shapes for membership functions.

Reasoning Under Uncertainty

Human expertise is based on effective application of learned biases. These biases must be tempered with an understanding of strengths and weaknesses (range of applicability) of each bias.
In expert systems, a model of inexact reasoning is needed to capture the judgmental, ``art of good guessing'' quality of science.
In this section we discuss several approaches to reasoning under uncertainty.

- Bayesian model of conditional probability
- EMYCIN's method, an approximation of Bayesian
- Bayesian nets, a more compact representation used for multiple variables.

UNIT-03/Lecture 04

Certainty Factors

Logic and rules provide all or nothing answers
An expert might want to say that something provides evidence for a conclusion, but it is not definite.
For example, the MYCIN system, an early expert system that diagnosed bacterial blood infections, used rules of this form:

if the infection is primary-bacteremia
and the site of the culture is one of the sterile sites
and the suspected portal of entry is the gastrointestinal tract
then there is suggestive evidence (0.7) that the infection is bacteroid

0.7 is a certainty factor

Certainty factors have been quantified using various different systems, including linguistics ones (certain, fairly certain, likely, unlikely, highly unlikely, definitely not) and various numeric scales, such as 0-10, 0-1, and -1 to 1. We shall concentrate on the -1 to 1 version.
Certainty factors may apply both to facts and to rules, or rather to the conclusion(s) of rules.
A "Theory" of Certainty
Certainty factors range from -1 to +1
As the certainty factor (CF) approaches 1 the evidence is stronger for a hypothesis.
As the CF approaches -1 the confidence against the hypothesis gets stronger.
A CF around 0 indicates that there is little evidence either for or against the hypothesis.
Certainty Factors and Rules
Premises for rules are formed by the and and or of a number of facts.
The certainty factors associated with each condition are combined to produce a certainty factor for the whole premise.
For two conditions P1 and P2:
CF(P1 and P2) = min(CF(P1), CF(P2))
CF(P1 or P2) = max(CF(P1), CF(P2))

The combined CF of the premises is then multiplied by the CF of the rule to get the CF of the conclusion
Example
if (P1 and P2) or P3 then C1 (0.7) and C2 (0.3)
Assume CF(P1) = 0.6, CF(P2) = 0.4, CF(P3) = 0.2
CF(P1 and P2) = min(0.6, 0.4) = 0.4
CF(0.4, P3) = max(0.4, 0.2) = 0.4
CF(C1) = 0.7 * 0.4 = 0.28
CF(C2) = 0.3 * 0.4 = 0.12

Combining Multiple CF's
Suppose two rules make conclusions about C.
How do we combine evidence from two rules?
Let CFR1(C) be the current CF for C.
Let CFR2(C) be the CF for C resulting from a new rule.
The new CF is calculated as follows:

CFR1(C) + CFR2(C) - CFR1(C) * CFR2(C) when CFR1(C)
and CFR2(C)
are both positive

CFR1(C) + CFR2(C) + CFR1(C) * CFR2(C) when CFR1(C)
and CFR2(C)
are both negative

[CFR1(C) + CFR2(C)]/[1 - min(|CFR1(C)|, |CFR2(C)|)] when CFR1(C)
and CFR2(C)
are of opposite sign

What do certainty factors mean?
- They are guesses by an expert about the relevance of evidence.
- They are ad hoc.
- CF's are tuned by trial and error.
- CF's hide more knowledge.

Certainty factors quantify the confidence that an expert might have in a conclusion that s/he has arrived at. We have given rules for combining certainty factors to obtain estimates of the certainty to be associated with conclusions obtained by using uncertain rules and uncertain evidence.

Certainty Factor : A certainty factor is a number, often in the range -1 to +1, which is associated with a condition or an action of a
rule. In more detail, each component of a condition may have an certainty factor associated with it - for example if the condition is of the form A and B, then there could be a certainty factor for A and a certainty factor for B.
A certainty factor of 1 means that the fact (or proposition) is highly certain. A certainty factor of 0 means no information about whether the proposition is true or not. A certainty factor of -1 means that the proposition is certainly false. A certainty factor of 0.7 means that the proposition is quite likely to be true, and so on.
The certainty factors of conditions are associated with facts held in working memory. Certainty factors for actions are stored as part of the rules.
Rules for manipulating certainty factors are given in the lecture notes on uncertain reasoning.
However, here is a simple example. Suppose that there is a rule
if P then Q (0.7)
meaning that if P is true, then, with certainty factor 0.7, Q follows. Suppose also that P is stored in working memory with an associated certainty factor of 0.8. Suppose that the rule above fires (see also match-resolve-act cycle). Then Q will be added to working memory with an associated certainty factor of 0.7 * 0.8 = 0.56.
condition-action rule
A condition-action rule, also called a production or production rule, is a rule of the form
if condition then action.
The condition may be a compound one using connectives like and, or, and not. The action, too, may be compound. The action can affect the value of
working memory variables, or take some real world action, or potentially do other things, including stopping the production system.
Rule-Based Systems
The knowledge of many expert systems is principally stored in their collections of rules.
One of the most popular methods for representing knowledge is in the form of Production Rules. These are in the form of:
if conditions then conclusion
If 1) the gram stain of the organism is gram
negative, and
2) the morphology of the organism is rod, and
3) the aerobicity of the organism is
anaerobic,
Then: There is suggestive evidence (. 6) that
the identity of the organism is
Bacteroides.

Advantages of Rules
• Knowledge comes in meaningful chunks.
• New knowledge can be added incrementally.
• Rules can make conclusions based on different kinds of data, depending on what is available.
• Rule conclusions provide ``islands'' that give multiplicative power.
• Rules can be used to provide explanations, control problem-solving process, check new rules for errors.
EMYCIN
EMYCIN was the first widely used expert system tool.
• Good for learning expert systems
• Limited in applicability to ``finite classification'' problems:
o Diagnosis
o Identification
• Good explanation capability
• Certainty factors

Several derivative versions exist.
Rule-Based Expert Systems[Shortliffe, E. Computer-based medical consultations: MYCIN. New York: Elsevier, 1976.]
MYCIN diagnoses infectious blood diseases using a backward-chained (exhaustive) control strategy.
The algorithm, ignoring certainty factors, is basically back chaining:
Given:
1. list of diseases, Goal-list
2. initial symptoms, DB
3. Rules

For each g $\in$ Goal-list do

If prove(g, DB, Rules) then Print (``Diagnosis:'', g)
Function prove (goal, DB, Rules)

If goal $\in$ DB then return True

elseif $\exists$ r $\in$ Rules such that rRHS contains goal

then return provelist(LHS, DB, Rules)[provelist calls prove with each condition of LHS]
else Ask user about goal and return answer
SLOT AND FILLER STRUCTURE
Why use this data structure?
• It enables attribute values to be retrieved quickly
o assertions are indexed by the entities
o binary predicates are indexed by first argument. E.g. team(Mike-Hall , Cardiff).
• Properties of relations are easy to describe .
• It allows ease of consideration as it embraces aspects of object oriented programming.
So called because:
• A slot is an attribute value pair in its simplest form.
• A filler is a value that a slot can take -- could be a numeric, string (or any data type) value or a pointer to another slot.
• A weak slot and filler structure does not consider the content of the representation.
We will study two types:
• Semantic Nets.
• Frames.

UNIT-03/Lecture 04

Semantic Network :
Semantic networks are a knowledge representation technique. More specifically, it is a way of recording all the relevant relationships between members of set of objects and types. "Object" means an individual (a particular person, or other particular animal or object, such as a particular cat, tree, chair, brick, etc.). "Type" means a set of related objects - the set of all persons, cats, trees, chairs, bricks, mammals, plants, furniture, etc. Possible relationships include the special set-theoretic relationships isa (set membership) and ako(the subset relation), and also general relationships like likes, child-of. Technically a semantic network is a node- and edge-labelled directed graph, and they are frequently depicted that way. Here is a pair of labelled nodes and a single labelled edge (relationship) between them (there could be more than one relationship between a single pair):

Here is a larger fragment of a semantic net, showing 4 labelled nodes (Fifi, cat, mammal, milk) and three labelled edges (isa, ako, likes) between them.

slot : A slot in a frame is like a field in a record or struct in languages like Pascal, Modula-2 and C.

However, slots can be added dynamically to frames, and slots contain substructure, called facets. The facets would normally include a value, perhaps a default, quite likely some demons, and possibly some flags like the iProlog frame system's cache and multi_valued facets.
state
The major idea is that:
• The meaning of a concept comes from its relationship to other concepts, and that,
• The information is stored by interconnecting nodes with labelled arcs.
Representation in a Semantic Net

These values can also be represented in logic as: isa(person, mammal), instance(Mike-Hall, person) team(Mike-Hall, Cardiff)
We have already seen how conventional predicates such as lecturer(dave) can be written as instance (dave, lecturer) Recall that isa and instance represent inheritance and are popular in many knowledge representation schemes. But we have a problem: How we can have more than 2 place predicates in semantic nets? E.g. score(Cardiff, Llanelli, 23-6) Solution:
• Create new nodes to represent new objects either contained or alluded to in the knowledge, game and fixture in the current example.
As a more complex example consider the sentence: John gave Mary the book. Here we have several aspects of an event.

Inference in a Semantic Net
Basic inference mechanism: follow links between nodes.
Two methods to do this:
Intersection search
-- the notion that spreading activation out of two nodes and finding their intersection finds relationships among objects. This is achieved by assigning a special tag to each visited node.
Many advantages including entity-based organisation and fast parallel implementation. However very structured questions need highly structured networks.
Inheritance
-- the isa and instance representation provide a mechanism to implement this.
Inheritance also provides a means of dealing with default reasoning. E.g. we could represent:
• Emus are birds.
• Typically birds fly and have wings.
• Emus run.
in the following Semantic net:

In making certain inferences we will also need to distinguish between the link that defines a new entity and holds its value and the other kind of link that relates two existing entities. Consider the example shown where the height of two people is depicted and we also wish to compare them.
We need extra nodes for the concept as well as its value.

Special procedures are needed to process these nodes, but without this distinction the analysis would be very limited.

Extending Semantic Nets
Here we will consider some extensions to Semantic nets that overcome a few problems (see Exercises) or extend their expression of knowledge.
Partitioned Networks Partitioned Semantic Networks allow for:
• propositions to be made without commitment to truth.
• expressions to be quantified.
Basic idea: Break network into spaces which consist of groups of nodes and arcs and regard each space as a node.
Consider the following: Andrew believes that the earth is flat. We can encode the proposition the earth is

flat in a space and within it have nodes and arcs the represent the fact (Fig. 15). We can the have nodes and arcs to link this space the the rest of the network to represent Andrew's belief.
Fig. 12 Partitioned network
Now consider the quantified expression: Every parent loves their child To represent this we:
• Create a general statement, GS, special class.
• Make node g an instance of GS.
• Every element will have at least 2 attributes:
o a form that states which relation is being asserted.
o one or more forall ( ) or exists ( ) connections -- these represent universally quantifiable variables in such statements e.g. x, y in parent(x) : child(y) loves(x,y)

Here we have to construct two spaces one for each x,y. NOTE: We can express variables as existentially qualified variables and express the event of love having an agent p and receiver b for every parent p which could simplify the network (See Exercises).
Also If we change the sentence to Every parent loves child then the node of the object being acted on (the child) lies outside the form of the general statement. Thus it is not viewed as an existentially qualified variable whose value may depend on the agent. (See Exercises and Rich and Knight book for examples of this) So we could construct a partitioned network as in Fig. 16

Fig. 12 Partitioned network
Frames : Frames are a knowledge representation technique. They resemble an extended form of record (as in Pascal and Modula-2) or struct (using C terminology) or class (in Java) in that they have a number of slots which are like fields in a record or struct, or variable in a class. Unlike a record/struct/class, it is possible to add slots to a frame dynamically (i.e. while the program is executing) and the contents of the slot need not be a simple value. If there is no value present in a slot, the frame system may use a default for frames of that type, or there may be a demon present to help compute a value for the slot.

Generic Frame : A frame that serves as a template for building instance frames. For example, a generic frame might describe the "elephant" concept in general, giving defaults for various elephant features (number of legs, ears, presence of trunk and tusks, colour, size, weight, habitat, membership of the class of mammals, etc.), which an instance frame would describe a particular elephant, say "Dumbo", who might have a missing tusk and who would thus have the default for number of tusks overridden by specifically setting number of tusks to 1. Instance frames are said to inherit their slots from the generic frame used to create them. Generic frames may also inherit slots from other generic frames of which they are a subconcept (as with mammal and elephant - elephant inherits all the properties of mammal that are encoded in the mammal generic frame - warm blood, bear young alive, etc.)
Goal state
Frames can also be regarded as an extension to Semantic nets. Indeed it is not clear where the distinction between a semantic net and a frame ends. Semantic nets initially we used to represent labelled connections between objects. As tasks became more complex the representation needs to be more structured. The more structured the system it becomes more beneficial to use frames. A frame is a collection of attributes or slots and associated values that describe some real world entity. Frames on their own are not particularly helpful but frame systems are a powerful way of encoding information to support reasoning. Set theory provides a good basis for understanding frame systems. Each frame represents:
• a class (set), or
An instFrame Knowledge Representation

Figure: A simple frame system
Here the frames Person, Adult-Male, Rugby-Player and Rugby-Team are all classes and the frames Robert-Howley and Cardiff-RFC are instances.
• The isa relation is in fact the subset relation.
• The instance relation is in fact element of.

• The isa attribute possesses a transitivity property. This implies: Robert-Howley is a Back and a Back is a Rugby-Player who in turn is an Adult-Male and also a Person.
• Both isa and instance have inverses which are called subclasses or all instances.
• There are attributes that are associated with the class or set such as cardinality and on the other hand there are attributes that are possessed by each member of the class or set.

DISTINCTION BETWEEN SETS AND INSTANCES

It is important that this distinction is clearly understood.
Cardiff-RFC can be thought of as a set of players or as an instance of a Rugby-Team.
If Cardiff-RFC were a class then
• its instances would be players
• it could not be a subclass of Rugby-Team otherwise its elements would be members of Rugby-Team which we do not want.

Instead we make it a subclass of Rugby-Player and this allows the players to inherit the correct properties enabling us to let the Cardiff-RFC to inherit information about teams.
This means that Cardiff-RFC is an instance of Rugby-Team.
BUT There is a problem here:
• A class is a set and its elements have properties.
• We wish to use inheritance to bestow values on its members.
• But there are properties that the set or class itself has such as the manager of a team.

This is why we need to view Cardiff-RFC as a subset of one class players and an instance of teams. We seem to have a CATCH 22. Solution: MetaClasses
A metaclass is a special class whose elements are themselves classes.
Now consider our rugby teams as:

Figure: A Metaclass frame system

The basic metaclass is Class, and this allows us to
• define classes which are instances of other classes, and (thus)
• inherit properties from this class.

Inheritance of default values occurs when one element or class is an instance of a class.

Slots as Objects

How can we to represent the following properties in frames?
• Attributes such as weight, age be attached and make sense.
• Constraints on values such as age being less than a hundred
• Default values
• Rules for inheritance of values such as children inheriting parent's names
• Rules for computing values
• Many values for a slot.

A slot is a relation that maps from its domain of classes to its range of values.
A relation is a set of ordered pairs so one relation is a subset of another.
Since slot is a set the set of all slots can be represent by a metaclass called Slot, say.
NOTE the following:
• Instances of SLOT are slots
• Associated with SLOT are attributes that each instance will inherit.
• Each slot has a domain and range.
• Range is split into two parts one the class of the elements and the other is a constraint which is a logical expression if absent it is taken to be true.
• If there is a value for default then it must be passed on unless an instance has its own value.
• The to-compute attribute involves a procedure to compute its value. E.g. in Position where we use the dot notation to assign values to the slot of a frame.
• Transfers through lists other slots from which values can be derived from inheritance.
• instance (an element of a class).

Interpreting frames

A frame system interpreter must be capable of the following in order to exploit the frame slot

representation:
• Consistency checking -- when a slot value is added to the frame relying on the domain attribute and that the value is legal using range and range constraints.
• Propagation of definition values along isa and instance links.
• Inheritance of default. values along isa and instance links.
• Computation of value of slot as needed.
• Checking that only correct number of values computed.
• Demon : A demon is a facet of a slot in a frame which causes some action to be taken when the frame is accessed in certain types of ways. For example, an if-needed demon is activated or triggered if the value of the slot is required and a value has not yet been stored in the slot, and it should calculate or otherwise obtain a value for the slot, while a range demon is triggered if a new value is added to the slot, to check that the value added is permissible for this particular slot.
• Here is a list of the demon types supported by the iProlog frame implementation:
if_added
demons are triggered when a new value is put into a slot.
if_removed
demons are triggered when a value is removed from a slot.
if_replaced
is triggered when a slot value is replaced.
if_needed
demons are triggered when there is no value present in an instance frame and a value must be computed from a generic frame.
if_new
is triggered when a new frame is created.
range
is triggered when a new value is added. The value must satisfy the range constraint specified for the slot.
help
is triggered when the range demon is triggered and returns false.
The following are not demons but demon-related slots in a frame.
cache
• means that when a value is computed it is stored in the instance frame.
• multi_valued
• means that the slot may contain more than one value.

Strong Slot and Filler Structures : Represent links between objects according to more rigid rules.
• Specific notions of what types of object and relations between them are provided.
• Represent knowledge about common situations.

UNIT-03/Lecture 05

Conceptual Dependency (CD)
Conceptual Dependency originally developed to represent knowledge acquired from natural language input.
The goals of this theory are:
• To help in the drawing of inference from sentences.
• To be independent of the words used in the original input.
• That is to say: For any 2 (or more) sentences that are identical in meaning there should be only one representation of that meaning.
It has been used by many programs that portend to understand English (MARGIE, SAM, PAM). CD developed by Schank et al. as were the previous examples.
CD provides: a structure into which nodes representing information can be placed

• a specific set of primitives
• at a given level of granularity.
Sentences are represented as a series of diagrams depicting actions using both abstract and real physical situations.
• The agent and the objects are represented
• The actions are built up from a set of primitive acts which can be modified by tense.
Examples of Primitive Acts are:
ATRANS -- Transfer of an abstract relationship. e.g. give.
PTRANS -- Transfer of the physical location of an object. e.g. go.
PROPEL -- Application of a physical force to an object. e.g. push.
MTRANS-- Transfer of mental information. e.g. tell.
MBUILD -- Construct new information from old. e.g. decide.
SPEAK -- Utter a sound. e.g. say.
ATTEND-- Focus a sense on a stimulus. e.g. listen, watch.
MOVE -- Movement of a body part by owner. e.g. punch, kick.
GRASP-- Actor grasping an object. e.g. clutch.
INGEST-- Actor ingesting an object. e.g. eat.
EXPEL -- Actor getting rid of an object from body. e.g. ????.
Six primitive conceptual categories provide building blocks which are the set of allowable dependencies in the concepts in a sentence:
PP-- Real world objects.
ACT-- Real world actions.
PA -- Attributes of objects.
AA -- Attributes of actions.
T-- Times.
LOC -- Locations.
How do we connect these things together?
Consider the example:
John gives Mary a book

• Arrows indicate the direction of dependency. Letters above indicate certain relationships:
o-- object.
R-- recipient-donor.
I -- instrument e.g. eat with a spoon.
D-- destination e.g. going home.
• Double arrows ( ) indicate two-way links between the actor (PP) and action (ACT).
• The actions are built from the set of primitive acts (see above).
o These can be modified by tense etc.
The use of tense and mood in describing events is extremely important and schank introduced the following modifiers:
p -- past
f-- future
t-- transition

-- start transition

-- finished transition
k
-- continuing
?
-- interrogative
/
-- negative

delta
-- timeless
c
-- conditional
the absence of any modifier implies the present tense.
So the past tense of the above example:
John gave Mary a book becomes:

The has an object (actor), PP and action, ACT. I.e. PP ACT. The triplearrow ( ) is also a two link but between an object, PP, and its attribute, PA. I.e. PP PA.
It represents isa type dependencies. E.g
Dave lecturerDave is a lecturer.
Primitive states are used to describe many state descriptions such as height, health, mental state, physical state.
There are many more physical states than primitive actions. They use a numeric scale.
E.g. John height(+10) John is the tallest John height(< average) John is short Frank Zappa health(-10) Frank Zappa is dead Dave mental_state(-10) Dave is sad Vase physical_state(-10) The vase is broken
You can also specify things like the time of occurrence in the relation ship.
For Example: John gave Mary the book yesterday

Now let us consider a more complex sentence: Since smoking can kill you, I stopped Lets look at how we represent the inference that smoking can kill:
• Use the notion of one to apply the knowledge to.
• Use the primitive act of INGESTing smoke from a cigarette to one.
• Killing is a transition from being alive to dead. We use triple arrows to indicate a transition from one state to another.
• Have a conditional, c causality link. The triple arrow indicates dependency of one concept on another.

To add the fact that I stopped smoking
• Use similar rules to imply that I smoke cigarettes.
• The qualification attached to this dependency indicates that the instance INGESTing smoke has stopped.

Advantages of CD:
• Using these primitives involves fewer inference rules.
• Many inference rules are already represented in CD structure.
• The holes in the initial structure help to focus on the points still to be established.
Disadvantages of CD:
• Knowledge must be decomposed into fairly low level primitives.
• Impossible or difficult to find correct set of primitives.
• A lot of inference may still be required.
• Representations can be complex even for relatively simple actions. Consider:
Dave bet Frank five pounds that Wales would win the Rugby World Cup.
Complex representations require a lot of storage

UNIT-03/Lecture 06

Scripts

A script is a structure that prescribes a set of circumstances which could be expected to follow on from one another. It is similar to a thought sequence or a chain of situations which could be anticipated.
Scripts are beneficial because:

- Events tend to occur in known runs or patterns.
- Causal relationships between events exist.
- Entry conditions exist which allow an event to take place
- Prerequisites exist upon events taking place. E.g. when a student progresses through a degree scheme or when a purchaser buys a house.

The components of a script include:
Entry Conditions -- these must be satisfied before events in the script can occur.
Results -- Conditions that will be true after events in script occur.
Props -- Slots representing objects involved in events.
Roles -- Persons involved in the events.
Track -- Variations on the script. Different tracks may share components of the same script.
Scenes-- The sequence of events that occur. Events are represented in conceptual dependency form.
Scripts are useful in describing certain situations such as robbing a bank. This might involve:

- Getting a gun.
- Hold up a bank.
- Escape with the money.

Here the Props might be

- Gun, G.
- Loot, L.
- Bag, B
- Get away car, C.

The Roles might be:

- Robber, S.
- Cashier, M.
- Bank Manager, O.
- Policeman, P.

The Entry Conditions might be:

- S is poor.
- S is destitute.

The Results might be:

- S has more money.
- O is angry.
- M is in a state of shock.
- P is shot.

There are 3 scenes: obtaining the gun, robbing the bank and the getaway.

- If a particular script is to be applied it must be activated and the activating depends on its significance.
- If a topic is mentioned in passing then a pointer to that script could be held.
- If the topic is important then the script should be opened.
- The danger lies in having too many active scripts much as one might have too many windows open on the screen or too many recursive calls in a program.
- Provided events follow a known trail we can use scripts to represent the actions involved and use them to answer detailed questions.
- Different trails may be allowed for different outcomes of Scripts (e.g. The bank robbery goes wrong).

CYC

What is CYC?

• An ambitious attempt to form a very large knowledge base aimed at capturing commonsense reasoning.
• Initial goals to capture knowledge from a hundred randomly selected articles in the EnCYClopedia Britannica.
• Both Implicit and Explicit knowledge encoded.
• Emphasis on study of underlying information (assumed by the authors but not needed to tell to the readers.
Example: Suppose we read that Wellington learned of Napoleon's death
Then we (humans) can conclude Napoleon never new that Wellington had died.
How do we do this?
We require special implicit knowledge or commonsense such as:
• We only die once.
• You stay dead.
• You cannot learn of anything when dead.
• Time cannot go backwards.
Why build large knowledge bases:
Brittleness
-- Specialised knowledge bases are brittle. Hard to encode new situations and non-graceful degradation in performance. Commonsense based knowledge bases should have a firmer foundation.
Form and Content
-- Knowledge representation may not be suitable for AI. Commonsense strategies could point out where difficulties in content may affect the form.
Shared Knowledge
-- Should allow greater communication among systems with common bases and assumptions.
Machine Learning : Machine learning refers to the ability of computers to automatically acquire new knowledge, learning from, for example, past cases or experience, from the computer's own experiences, or from exploration. Machine learning has many uses such as finding rules to direct marketing campaigns based on lessons learned from analysis of data from supermarket loyalty campaigns; or learning to recognize characters from people's handwriting. Machine learning enables computer software to adapt to changing circumstances, enabling it to make better decisions than non-AI software. Synonyms: learning, automatic learning.

Model-based Reasoning : Model-based reasoning (MBR) concentrates on reasoning about a system’s behavior from an explicit model of the mechanisms underlying that behavior. Model-based techniques can very succinctly represent knowledge more completely and at a greater level of detail than techniques that encode experience, because they employ models that are compact axiomatic systems from which large amounts of information can be deduced.

Natural Language Processing : English is an example of a natural language, a computer language isn't. For a computer to process a natural language, it would have to mimic what a human does. That is, the computer would have to recognize the sequence of words spoken by a person or another computer, understand the syntax or grammar of the words (i.e., do a syntactical analysis), and then extract the meaning of the words. A limited amount of meaning can be derived from a sequence of words taken out of context (i.e., by semantic analysis); but much more of the meaning depends on the context in which the words are spoken (e.g., who spoke them, under what circumstances, with what tone, and what else was said, particularly before the words), which would require a pragmatic analysis to extract. To date, natural language processing is poorly developed and computers are not yet able to even approach the ability of humans to extract meaning from natural languages; yet there are already valuable practical applications of the technology.

How is CYC coded?
• By hand.
• Special CYCL language:

o LISP like.
o Frame based
o Multiple inheritance
o Slots are fully fledged objects.
o Generalized inheritance -- any link not just isa and instance.

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.

If you have a choice of number of items k from a set of items n then the formula is applied to find the number of ways of making this choice. (! = factorial).

So the chance of winning the national lottery (choosing 6 from 49) is to 1.

• Conditional probability, P(A|B), indicates the probability of of event A given that we know event B has occurred.
• sets etc. and reason effectively.