7. Symbolic Reasoning under Uncertainty

We have described techniques for reasoning with a complete, consistent and unchanging model of the world. But in many problem domains, it is not possible to create such models. So here we are going to explore techniques for solving problems with incomplete and uncertain models.

What is reasoning?

- When we require any knowledge system to do something it has not been explicitly told how to do it must *reason.*
- The system must figure out what it needs to know from what it already knows.
    - Reasoning is the act of deriving a conclusion from certain premises using a given methodology. (Process of thinking/ Drawing inference)

How can we Reason?

- To a certain extent this will depend on the knowledge representation chosen.
- Although a good knowledge representation scheme has to allow easy, natural and plausible reasoning.

Broad Methods of how we may reason

- Formal reasoning –
    - Basic rules of inference with logic knowledge representations.
- Procedural reasoning –
    - Uses procedures that specify *how to perhaps solve (sub) problems.*
- Reasoning by analogy –
    - Humans are good at this, more difficult for AI systems. E.g. If we are asked *Can robins fly?. The system might reason that robins are like sparrows and it knows sparrows can fly so ...*
- Generalization and abstraction –
    - Again humans effective at this. This is basically getting towards *learning and understanding methods.*
- Meta-level reasoning –
    - *Once again uses knowledge about what you know and perhaps ordering it in some kind of importance.*

Uncertain Reasoning

- Unfortunately the world is an uncertain place.
- Any AI system that seeks to model and reasoning in such a world must be able to deal with this.
- In particular it must be able to deal with:
    - Incompleteness – compensate for lack of knowledge.

    - Inconsistencies – resolve ambiguities and contradictions.
    - Change – it must be able to update its *world knowledge base over time.*
- Clearly in order to deal with this some decision that a made are more likely to be true (or false) than others and we must introduce methods that can cope with this *uncertainty.*

Monotonic Reasoning

Predicate logic and the inferences we perform on it is an example of monotonic reasoning. In monotonic reasoning if we enlarge at set of axioms we cannot retract any existing assertions or axioms.

A monotonic logic cannot handle
- Reasoning by default
    - Because consequences may be derived only because of lack of evidence of the contrary
- Abductive Reasoning
    - Because consequences are only deduced as most likely explanations.
- Belief Revision
    - Because new knowledge may contradict old beliefs.

Non-Monotonic Reasoning

- Non monotonic reasoning is one in which the axioms and/or the rules of inference are extended to make it possible to reason with incomplete information. These systems preserve, however, the property that, at any given moment, a statement is either
    - believed to be true,
    - believed to be false, or
    - not believed to be either.
- Statistical Reasoning: in which the representation is extended to allow some kind of numeric measure of certainty (rather than true or false) to be associated with each statement.
- In a system doing *non-monotonic reasoning* the set of conclusions may either grow or shrink when new information is obtained.
- Non-monotonic logics are used to formalize plausible reasoning, such as the following inference step:

```
Birds typically fly.
Tweety is a bird.
-------------------------
Tweety (presumably) flies.
```

- Such reasoning is characteristic of commonsense reasoning, where *default rules* are applied when case-specific information is not available. The conclusion of non-monotonic argument may turn out to be wrong. For example, `if Tweety is a penguin, it is incorrect to conclude that Tweety flies.`
- Non-monotonic reasoning often requires jumping to a conclusion and subsequently retracting that conclusion as further information becomes available.
- All systems of non-monotonic reasoning are concerned with the issue of consistency.

- Inconsistency is resolved by removing the relevant conclusion(s) derived previously by default rules.
- Simply speaking, the truth value of propositions in a nonmonotonic logic can be classified into the following types:
    - *facts* that are definitely true, such as "Tweety is a bird"
    - *default rules* that are normally true, such as "Birds fly"
    - *tentative conclusions* that are presumably true, such as "Tweety flies"
- When an inconsistency is recognized, only the truth value of the last type is changed.

Properties of FOPL

- It is complete with respect to the domain of interest.
- It is consistent.
- The only way it can change is that new facts can be added as they become available.
    - If these new facts are consistent with all the other facts that have already have been asserted, then nothing will ever be retracted from the set of facts that are known to be true.
    - This is known as “monotonicity”.

If any of these properties is not satisfied, conventional logic based reasoning systems become inadequate.

Non monotonic reasoning systems, are designed to be able to solve problems in which all of these properties may be missing Issues to be addressed:

- How can the knowledge base be extended to allow inferences to be made on the basis of lack of knowledge as well as on the presence of it?
    - We need to make clear the distinction between
        - It is known that P.
        - It is not known whether P.
    - First-order predicate logic allows reasoning to be based on the first of these. ○ In our new system, we call any inference that depends on the lack of some piece of knowledge a non-monotonic inference.
    - Traditional systems based on predicate logic are monotonic. Here number of statements known to be true increases with time.
    - New statements are added and new theorems are proved, but the previously known statements never become invalid.
- How can the knowledge base be updated properly when a new fact is added to the system(or when the old one is removed)?
    - In Non-Monotonic systems, since addition of a fact can cause previously discovered proofs to become invalid,
        - how can those proofs, and all the conclusions that depend on them be found?
        - Solution: keep track of proofs, which are often called justifications.
    - Such a recording mechanism also makes it possible to support,

        - monotonic reasoning in the case where axioms must occasionally be retracted to reflect changes in the world that is being modeled.
- How can knowledge be used to help resolve conflicts when there are several in consistent non monotonic inferences that could be drawn?
    - It turns out that when inferences can be based
        - on the lack of knowledge as well as on its presence,
        - contradictions are much likely to occur than they were in conventional logical systems in which the only possible contradictions.

Default Reasoning

- Non monotonic reasoning is based on default reasoning or "most probabilistic choice".
    - S is assumed to be true as long as there is no evidence to the contrary.
- Default reasoning (or most probabilistic choice) is defined as follows:
    - Definition 1 : If X is not known, then conclude Y.
    - Definition 2 : If X can not be proved, then conclude Y.
    - Definition 3: If X can not be proved in some allocated amount of time then conclude Y.

Logics for Non-Monotonic Reasoning

- Monotonicity is fundamental to the definition of first-order predicate logic, we are forced to find some alternative to support non-monotonic reasoning.
- We examine several because no single formalism with all the desired properties has yet emerged.
- We would like to find a formalism that does all of the following things:
    - Defines the set of possible worlds that could exist given the facts that we do have.
    - More precisely, we will define an interpretation of a
        - set of wff's to be a domain (a set of objects), together with a function that assigns; to each predicate, a relation;
        - to each n-ary function, an operator that maps from D'' to D; and to each constant, an element of D.
    - A model of a set of wff's is an interpretation that satisfies them.
- Provides a way to say that we prefer to believe in some models rather than others.
- Provides the basis for a practical implementation of this kind of reasoning.
- Corresponds to our intuitions about how this kind of reasoning works.

Default Reasoning

- This is a very common form of non-monotonic reasoning.
- Here we want to draw conclusions based on *what is most likely to be true*.
- Two Approaches to do this
    - Non-Monotonic Logic

    - Default Logic
- Non-Monotonic reasoning is generic descriptions of a class of reasoning.
- Non-Monotonic logic is a specific theory.
- The same goes for Default reasoning and Default logic.

Non-monotonic Logic

- One system that provides a basis for default reasoning is Non-monotonic Logic (NML).
- This is basically an extension of first-order predicate logic to include a modal operator, M.
    - The purpose of this is to allow for consistency.

$$\forall x: \text{plays\_instrument}(x) \wedge M\ \text{improvises}(x) \rightarrow \text{jazz\_musician}(x)$$

states that

- for all *x* is *x* plays an instrument and if the fact that *x* can improvise is consistent with all other knowledge
- then we can conclude that *x* is a jazz musician.

$$\forall x,y: \text{related}(x,y) \wedge M\ \text{GetAlong}(x,y) \rightarrow \text{WillDefend}(x,y)$$

states that

- for all *x* and y, if x and y are related and if the fact that x gets along with y is consistent with everything else that is beleieved,
- then we can conclude that x will defend y

How do we define *consistency*?

One common solution (consistent with PROLOG notation) is to show that fact *P* is true attempt to prove. If we fail we may say that *P* is consistent (since is false). However consider the famous set of assertions relating to President Nixon.

$$\forall x: \text{Republican}(x) \wedge M \neg\text{Pacifist}(x) \rightarrow \neg\text{Pacifist}(x)$$

$$\forall x: \text{Quaker}(x) \wedge M\ \text{Pacifist}(x) \rightarrow \text{Pacifist}(x)$$

Now this states that Quakers tend to be pacifists and Republicans tend not to be. BUT Nixon was both a Quaker and a Republican so we could assert:

Quaker(Nixon)

Republican(Nixon)

This now leads to our total knowledge becoming inconsistent.

What conclusions does the theory actually support?

NML defines the set of theorems that can be derived from a set of wff's A to be the intersection of the sets of theorems that result from the various ways in which the wff's of A might be combined.

Default Logic

An alternative logic for performing default based reasoning is Reiter's Default Logic (DL). Default logic introduces a new inference rule of the form:

$$\frac{A : B}{C}$$

which states if A is provable and it is consistent to assume B then conclude C.

Now this is similar to Non-monotonic logic but there are some distinctions:

New inference rules are used for computing the set of plausible extensions. So in the Nixon example above Default logic can support both assertions since is does not say anything about how choose between them -- it will depend on the inference being made. In Default logic any nonmonotonic expressions are rules of inference rather than expressions. They cannot be manipulated by the other rules of inference. This leads to some unexpected results.

In Default Logic, A indicates prerequisite, B indicates justification, and C indicates Consequence.

Example: Typically "An American adult owns a car."

$$\frac{American(x) \wedge Adult(x) : M(\exists y\ car(y) \wedge owns(x, y))}{\exists y\ car(y) \wedge owns(x, y)}$$

If we can prove from our beliefs that x is American and adult and believing that there is some car that is owned by x does not lead to an inconsistency.

Inheritance:

One very common use of nonmonotonic reasoning is as a basis for inheriting attribute values from a prototype description of a class to the individual entities that belong to the class. Considering the Baseball example in Inheritable Knowledge, and try to write its inheriting knowledge as rules in DL.

We can write a rule to account for the inheritance of a default value for the height of a baseball player as:

$$\frac{Baseball\text{-}Player(x) : height(x, 6\text{-}1)}{height(x, 6\text{-}1)}$$

Now suppose we assert Pitcher(Three-Finger-Brown). Since this enables us to conclude that Three-Finger-Brown is a baseball player, our rule allows us to conclude that his height is 6-1. If, on the other hand, we had asserted a conflicting value for Three Finger had an axiom like:

$$\forall x, y, z : height(x, y) \wedge height(x, z) \rightarrow y = z,$$

Which prohibits someone from having more than one height, then we would not be able to apply the default rule. Thus an explicitly stated value will block the inheritance of a default value, which is exactly what we want.

Let's encode the default rule for the height of adult males in general. If we pattern it after the one for baseball players, we get

$$\frac{Adult\text{-}Male(x) : height(x, 5\text{-}10)}{height(x, 5\text{-}10)}$$

Unfortunately, this rule does not work as we would like. In particular, if we again assert Pitcher(Three-Finger-Brown) then the resulting theory contains 2 extensions: one in which our first rule fires and brown's height is 6-1 and one in which this rule applies and Brown's height is 510. Neither of these extensions is preferred. In order to state that we prefer to get a value from the more specific category, baseball player, we could rewrite the default rule for adult males in general as:

$$\frac{Adult\text{-}Male(x) : \neg Baseball\text{-}Player(x) \wedge height(x, 5\text{-}10)}{height(x, 5\text{-}10)}$$

This effectively blocks the application of the default knowledge about adult males in this case that more specific information from the class of baseball players is available. Unfortunately, this approach can become widely as the set of exceptions to the general rule increases. We would end up with a rule like:

$$\frac{Adult\text{-}Male(x) : \neg Baseball\text{-}Player(x) \wedge \neg Midget(x) \wedge - Jockey(x) \wedge height(x, 5\text{-}10)}{height(x, 5\text{-}10)}$$

A clearer approach is to say something like. Adult males typically have a height of 5-10 unless they are abnormal in some way. We can then associate with other classes the information that they are abnormal in one or another way. So we could write, for example:

$\forall x$: $Adult\text{-}Male(x) \wedge \neg AB(x, aspect1) \rightarrow height(x, 5\text{-}10)$
$\forall x : Baseball\text{-}Player(x) \rightarrow AB(x, aspect\ 1)$
$\forall x : Midget(x) \rightarrow AB(x, aspect\ 1)$
$\forall x : Jockey(x) \rightarrow AB(x, aspect\ 1)$

Then, if we add the single default rule:

$$\frac{: \neg AB(x, y)}{\neg AB(x, y)}$$

we get the desired result.

Abduction

Abductive reasoning is to abduce (or take away) a logical assumption, explanation, inference, conclusion, hypothesis, or best guess from an observation or set of observations. Because the conclusion is merely a best guess, the conclusion that is drawn may or may not be true. Daily decision-making is also an example of abductive reasoning.

Standard logic performs deductions. Given 2 axioms

- $\forall x : A(x) \rightarrow B(x)$, $A(C)$, We conclude $B(C)$ using deduction

$\forall x$: Measles(x) → Spots(x), Having measles implies having spots

If we notice Spots, we might like to conclude measles, but it may be wrong. But may be a best guess, we can make about what is going on. Deriving conclusions in this way is abductive reasoning (a form of default reasoning).

- Given 2 wff's (A→B) & (B), for any expression A & B, if it is consistent to assume A, do so.

Minimalist Reasoning

We describe methods for saying a very specific and highly useful class of things that are generally true. These methods are based on some variant of the idea of a minimal model. We will define a model to be minimal if there are no other models in which fewer things are true. The idea behind using minimal models as a basis for non-monotonic reasoning about the world is the following –

- *There are many fewer true statements than false ones.*
- *If something is true and relevant it makes sense to assume that it has been entered into our knowledge base.*
- *Therefore, assume that the only true statements are those that necessarily must be true in order to maintain the consistency.*

The Closed World Assumption

- CWA
  - Simple kind of minimalist reasoning.
  - says that the only objects that satisfy any predicate P are those that must.
  - Eg. A company's employee database, Airline example
- CWA is powerful as a basis for reasoning with Databases, which are assumed to be complete with respect to the properties they describe.
- Although the CWA is both simple & powerful, it can fail to produce an appropriate answer for either of the two reasons.
  - The assumptions are not always true in the world; some parts of the world are not realistically "closable". - unrevealed facts in murder case
  - It is a purely syntactic reasoning process. Thus, the result depends on the form of assertions that are provided.

Consider a KB that consists of just a single statement A(Joe) v B(Joe)

The CWA allows us to conclude both ?A(Joe) and ?B(Joe), since neither A nor B must necessarily be true of Joe.

The extended KB

(

$$A(Joe) \vee B(Joe)$$

$$\neg A(Joe)$$

$$\neg B(Joe)$$ is inconsistent.

The problem is that we have assigned a special status to the positive instances of predicates as opposed to negative ones. CWA forces completion of KB by adding negative assertion whenever it is consistent to do so.

CWA captures part of the idea that anything that must not necessarily be true should be assumed to be false, it does not capture all of it.

It has two essential limitations:

- It operates on individual predicates without considering interactions among predicates that are defined in the KB.
- It assumes that all predicates have all their instances listed. Although in many database applications this is true, in many KB systems it is not.

Circumscription

- *Circumscription* is a rule of conjecture (*conclusion formed on the basis of incomplete information*) that allows you
  - to jump to the conclusion that the objects you can show that posses a certain property, *p*, are in fact all the objects that posses that property.
- Circumscription can also cope with default reasoning. Several theories of circumscription have been proposed to deal with the problems of CWA.
- Circumscription together with first order logic allows a form of Non-monotonic Reasoning.

Suppose we know:

$$bird(tweety)$$
$$\forall x: penguin(x) \rightarrow bird(x)$$
$$\forall x: penguin(x) \rightarrow \neg flies(x)$$

And we wish to add the fact that *typically, birds fly*.

In circumscription this phrase would be stated as:

A bird will flyif it is not abnormal

and can thus be represented by:

$$\forall x: \quad ( ) \wedge \neg \quad ( ) \quad ' \quad ( ).$$

However, this is not sufficient. We cannot conclude

( ) since we cannot prove ( ).

This is where we apply circumscription and, in this case, *we will assume that those things that are shown to be abnormal are the only things to be abnormal.* Thus we can rewrite our *default rule* as:

$$\forall x: \quad ( ) \qquad \wedge \neg \qquad ( ) \neg \qquad ( )$$

and add the following

$$\forall x: \neg \qquad ( )$$

since there is nothing that cannot be shown to be abnormal.

If we now add the fact: ( ) Clearly we can prove ( ).

If we circumscribe abnormal now we would add the sentence, *penguin (tweety) is the abnormal thing*:

$$\forall x: \quad ( ) \rightarrow \quad ( ).$$

Note the distinction between Default logic and circumscription:

- *Defaults are sentences in language itself not additional inference rules*.

Dependency Directed Backtracking

- To reduce the computational cost of non-monotonic logic, we need to be able to avoid re-searching the entire search space when a new piece of evidence is introduced
  - otherwise, we have to backtrack to the location where our assumption was introduced and start searching anew from there
- In dependency directed backtracking, we move to the location of our assumption, make the change and propagate it forward from that point without necessarily having to research from scratch
  - as an example, you have scheduled a meeting on Tuesday at 12:15 because everyone indicated that they were available
  - but now, you cannot find a room, so you backtrack to the day and change it to Thursday, but you do not re-search for a new time because you assume if everyone was free on Tuesday, they will be free on Thursday as well

Implementations: Truth Maintenance Systems

A variety of *Truth Maintenance Systems* (TMS) have been developed as a means of implementing Non-Monotonic Reasoning Systems.

- tracking the order in which sentences are told to the knowledge base by numbering them, this implies that the KB will be consistent.

The idea of truth maintenance system arose as a way of providing the ability to do dependencydirected backtracking and so to support nonmonotonic reasoning.

Types of TMS:

- justification-based TMS (JTMS)
- assumption-based TMS (ATMS)
- logic-based TMS (LTMS)

Basically TMSs:

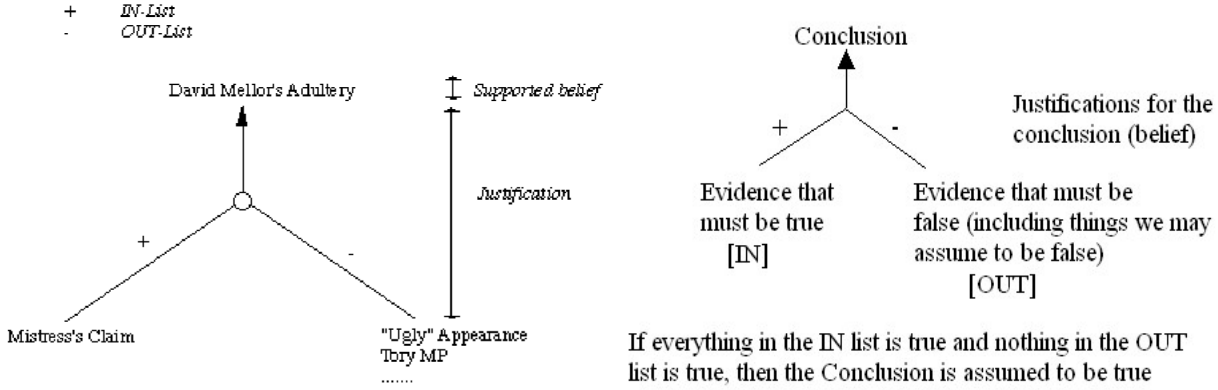
- all do some form of dependency directed backtracking Assertions are connected via a network of dependencies.

Justification-Based Truth Maintenance Systems (JTMS)

- This is a simple TMS in that it does not know anything about the structure of the assertions themselves.
- JTMS is one element of a TMS design space, a good model for most dependency systems and can quickly focus on how to use it.
- This TMS itself does not know anything about the structure of the assertions themselves.
- The only role is to serve as a bookkeeper for a separate problem solving system which in turn provides it with both assertions and dependencies among assertions.

How JTMS works?

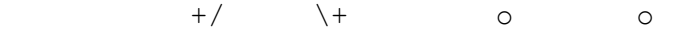
- Each supported belief (assertion) in has a justification.
- Each justification has two parts:
  - An *IN-List* -- which supports beliefs held.
  - An *OUT-List* -- which supports beliefs *not* held.
- An assertion is connected to its justification by an arrow.
- One assertion can *feed* another justification thus creating the network.
- Assertions may be labelled with a *belief status*.
- An assertion is *valid* if every assertion in the IN-List is believed and none in the OUT-List are believed.
- An assertion is non-monotonic is the OUT-List is not empty or if any assertion in the IN-List is non-monotonic.



A Justification-based truth maintenance system (JTMS) is a simple TMS where one can examine the consequences of the current set of assumptions. In JTMS labels are attached to arcs from sentence nodes to justification nodes. This label is either "+" or "-". Then, for a justification node we can talk of its *IN-LIST,* the list of its inputs with "+" label, and of its *OUT-LIST*, the list of its inputs with "-" label.

The meaning of sentences is not known. We can have a node representing a sentence p and one representing ~p and the two will be totally unrelated, unless relations are established between them by justifications. For example, we can write:

```
          ~p^p Contradiction Node
            o
|
            x      'x' denotes a justification node
          /   \          'o' denotes a sentence node
```



p ~p which says that if both p and ~p are IN we have a contradiction.

The association of IN or OUT labels with the nodes in a dependency network defines an *in-outlabeling function*. This function is consistent if:

- The label of a junctification node is IN iff the labels of all the sentence nodes in its in-list are all IN and the labels of all the sentence nodes in its out-list are OUT.
- The label of a sentence node is IN iff it is a premise, or an enabled assumption node, or it has an input from a justification node with label IN.

A set of important reasoning operations that a JTMS does not perform, including:

- Applying rules to derive conclusions
- Creating justifications for the results of applying rules
- Choosing among alternative ways of resolving a contradiction
- Detecting contradictions

All of these operations must be performed by the problem-solving program that is using the JTMS.

Logic-Based Truth Maintenance Systems (LTMS)
Similar to JTMS except:

- Nodes (assertions) assume no relationships among them except ones explicitly stated in justifications.
- JTMS can represent P and P IN simultaneously. No contradiction will be detected automatically. An LTMS would throw a contradiction automatically in such a case here.
- If this happens network has to be reconstructed.

Assumption-Based Truth Maintenance Systems (ATMS)

- JTMS and LTMS pursue a single line of reasoning at a time and backtrack (dependencydirected) when needed → *depth first search*.
- ATMS maintain alternative paths in parallel → *breadth-first search*
- Backtracking is avoided at the expense of maintaining multiple contexts.
- However as reasoning proceeds contradictions arise and the ATMS can be *pruned* ○ Simply find assertion with no valid justification.

The ATMS like the JTMS is designed to be used in conjunction with a separate problem solver. The problem solver's job is to:

- Create nodes that correspond to assertions (both those that are given as axioms and those that are derived by the problem solver).
- Associate with each such node one or more justifications, each of which describes reasoning chain that led to the node.
- Inform the ATMS of inconsistent contexts.

This is identical to the role of the problem solver that uses a JTMS, except that no explicit choices among paths to follow need to be made as reasoning proceeds. Some decision may be necessary at the end, though, if more than one possible solution still has a consistent context.

The role of the ATMS system is then to:

- Propagate inconsistencies, thus ruling out contexts that include subcontexts (set of assertions) that are known to be inconsistent.
- Label each problem solver node with the contexts in which it has a valid justification. This is done by combining contexts that correspond to the components of a justification. In particular, given a justification of the form

  1 2 … →

  assign as a context for the node corresponding to C the intersection of the contexts corresponding to the nodes A1 through An.

Contexts get eliminated as a result of the problem-solver asserting inconsistencies and the ATMS propagating them. Nodes get created by the problem-solver to represent possible components of a problem solution. They may then get pruned from consideration if all their context labels get pruned.

8. Statistical Reasoning

Introduction:

Statistical Reasoning: The reasoning in which the representation is extended to allow some kind of numeric measure of certainty (rather than true or false) to be associated with each statement. A fact is believed to be true or false. For some kind of problems, describe beliefs that are not certain but for which there is a supporting evidence.

There are 2 class of problems:

- First class contain problems in which there is genuine randomness in the word.
    - ○ Example: Cards Playing
- Second class contains problems that could in principle be modeled using the technique we described (i.e. resolution from predicate logic) ○ Example: Medical Diagnosis

Probability & Baye's Theorem

An important goal for many problem-solving systems is to collect evidence as the system goes along and to model its behavior on the basis of the evidence. To model this behavior, we need statistical theory of evidence. Bayesian statistics is such a theory. The fundamental notion of Bayesian statistics is that of conditional probability. Conditional Probability is the probability of an event given that another event has occurred.

Read this expression as the probability of hypothesis H given that we have observed evidence E. To compute this, we need to take into account the prior probability of H and the extent to which E provides evidence of H.

$$P(H_i \backslash E) = \frac{P(E \mid H_i) \cdot P(H_i)}{\sum_{n=1}^{k} P(E \mid H_n) \cdot P(H_n)}$$

- P(H/E) = probability of hypothesis H given that we have observed evidence E
- P(Hi/E) = probability of hypothesis Hi is true under the evidence E
- P(E/Hi) = probability that we will observe evidence E given that hypothesis Hi is true
- P(Hi) = a priori probability that hypothesis is true in absence of any specific evidence
- k = number of possible hypothesis

Suppose, for example, that we are interested in examining the geological evidence at a particular location to determine whether that would be a good place to dig to find a desired mineral. If we know the prior probabilities of finding each of the various minerals and we know the probabilities that if a mineral is present then certain physical characteristics will be observed, then we use the Baye's formula to compute from the evidence we collect, how likely it is that the various minerals are present.

The key to using Baye's theorem as a basis for uncertain reasoning is to recognize exactly what it says.

Consider the following puzzle:

> A pea is placed under one of three shells, and the shells are then manipulated in such a fashion that all three appear to be equally likely to contain the pea. Nevertheless, you win a prize if you guess the correct shell, so you make a guess. The person running the game does know the correct shell, however, and uncovers one of the shells that you did not choose and that is empty. Thus, what remains are two shells: one you chose and one you did not choose. Furthermore, since the uncovered shell did not contain the pea, one of the two remaining shells does contain it. You are offered the opportunity to change your selection to the other shell. Should you?

Work through the conditional probabilities mentioned in this problem using Bayes' theorem. What do the results tell about what you should do'

Let $P(G) =$ probability that my initial guess is right.
$P(O) =$ probability that one of the other 2 shells has the pea.
$P(T) =$ Probability that a shell without a pea will be turned over.

$P(G) = \frac{1}{3}$

$P(O) = \frac{2}{3}$

$P(T) = 1$

The important fact here is that $P(T) = 1$, i.e., it is certain that a shell without a pea will be turned over. So the turning over provides no new information. So:

$P(G|T) = P(G) = \frac{1}{3}$

$P(O|T) = P(O) = \frac{2}{3}$

But O now contains only one shell that hasn't been turned over. So the probability that that shell contains the pea is $\frac{2}{3}$. You should change your guess to that other shell.

To see what is really going on here, we can contrast this problem with one in which, after you make your guess, the other person turns over one of the other shells are random (i.e., without knowing where the pea is). Then:

$P(G) = \frac{1}{3}$

$P(O) = \frac{2}{3}$

$P(T) = \frac{2}{3}$

$P(G|T) = \frac{P(G \wedge T)}{P(T)} = \frac{\frac{1}{3}}{\frac{2}{3}} = \frac{1}{2}$

We got this by using the simplest form of Bayes rule and observing that if your guess was right, then the other person is certain to pick a shell without a pea, so $P(G \wedge T)$ is just $P(G)$.

Suppose we are solving a medical diagnosis problem. Consider the following assertions:

:

:

:

- Without any additional evidence, the presence of spots serves as evidence in favor of measles. It also serves as evidence of fever since measles would cause fever.
- Suppose we already know that the patient has measles. Then the additional evidence that he has spots actually tells us nothing about fever.
- Either spots alone or fever alone would constitute evidence in favor of measles.
- If both are present, we need to take both into account in determining the total weight of evidence.

In a clinic, the probability of the patients having HIV virus is **0.15**.

A blood test done on patients :

If patient has virus, then the test is **+ve** with probability **0.95**.

If the patient does not have the virus, then the test is **+ve** with probability **0.02**.

Assign labels to events : **H** = patient has virus; **P** = test +ve

Given : $P(H) = 0.15$; $P(P|H) = 0.95$; $P(P|\neg H) = 0.02$

Find :

If the test is **+ve** what are the probabilities that the patient

i) has the virus ie $P(H|P)$; ii) does not have virus ie $P(\neg H|P)$;

If the test is **-ve** what are the probabilities that the patient

iii) has the virus ie $P(H|\neg P)$; iv) does not have virus ie $P(\neg H|\neg P)$;

Calculations :

i) For $P(H|P)$ we can write down Bayes Theorem as

$P(H|P) = [\, P(P|H)\, P(H)\,] \,/\, P(P)$

We know $P(P|H)$ and $P(H)$ but not $P(P)$ which is probability of a **+ve** result.

There are two cases, that a patient could have a **+ve** result, stated below :

1. Patient has virus and gets a **+ve** result : $H \cap P$
2. Patient does not have virus and gets a **+ve** result: $\neg H \cap P$

Find probabilities for the above two cases and then add

Find probabilities for the above two cases and then add

ie $P(P) = P(H \cap P) + P(\neg H \cap P)$.

But from the second axiom of probability we have :

$P(H \cap P) = P(P|H)\, P(H)$ and $P(\neg H \cap P) = P(P|\neg H)\, P(\neg H)$.

Therefore putting these we get :

$$P(P) = P(P|H)\, P(H) + P(P|\neg H)\, P(\neg H) = 0.95 \times 0.15 + 0.02 \times 0.85 = 0.1595$$

Now substitute this into Bayes Theorem and obtain $P(H|P)$

$$P(H|P) = \frac{P(P|H)\, P(H)}{P(P|H)\, P(H) + P(P|\neg H)\, P(\neg H)} = 0.95 \times 0.15 \,/\, 0.1595 = 0.8934$$

ii) Next is to work out $P(\neg H|P)$

$$P(\neg H|P) = 1 - P(H|P) = 1 - 0.8934 = 0.1066$$

iii) Next is to work out $P(H|\neg P)$; again we write down Bayes Theorem

$$P(H|\neg P) = \frac{P(\neg P|H)\, P(H)}{P(\neg P)}$$ here we need $P(\neg P)$ which is $1 - P(P)$

$$= (0.05 \times 0.15)/(1-0.1595) = 0.008923$$

iv) Finally, work out $P(\neg H|\neg P)$

It is just $1 - P(H|\neg P) = 1 - 0.008923 = 0.99107$

Disadvantages with Baye's Theorem

- The size of set of joint probability that we require in order to compute this function grows as 2^n if there are n different propositions considered.
- Baye's Theorem is hard to deal with for several reasons:
    - Too many probabilities have to be provided
    - the space that would be required to store all the probabilities is too large.
    - time required to compute the probabilities is too large.

Mechanisms for making easy to deal with uncertain reasoning

- Attaching Certainty factor to rules
- Bayesian Networks
- Dempster-Shafer Theory
- Fuzzy Logic

Certainty Factors and Rule-Based Systems

The certainty-factor model was one of the most popular model for the representation and manipulation of uncertain knowledge in the early 1980s Rule-based expert systems. Expert systems are an example for the certainty factors.

We describe one practical way of compromising on a pure Bayesian system. MYCIN system is an example of an expert system, since it performs a task normally done by a human expert. MYCIN system attempts to recommend appropriate therapies for patients with bacterial infections. It interacts with the physician to acquire the clinical data it needs. We concentrate on the use of probabilistic reasoning.

MYCIN represents most of its diagnostic knowledge as a set of rules. Each rule has associated with it a certainty factor, which is a measure of the extent to which the evidence is described by antecedent of the rule, supports the conclusion that is given in the rule's consequent. It uses backward reasoning to the clinical data available from its goal of finding significant diseasecausing organisms.

What do Certainty Factor Mean?

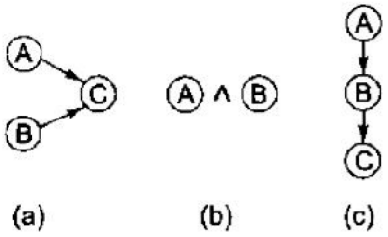
- It is an expert estimate of degree of belief or disbelief in an evidence hypothesis relation.
- A certainty factor (CF[h,e]) is defined in terms of two components ◦ MB [h, e]:
    - A measure between 0 & 1 of belief in hypothesis h given the evidence e.
    - MB measures the extent to which the evidence supports the hypothesis
    - MB=0, if the evidence fails to support hypothesis ◦ MD [h, e]:
    - A measure between 0 & 1 of disbelief in hypothesis h given by the evidence 'e'
    - MD measures the extent to which the evidence does not support hypothesis
    - MD=0, if the evidence supports the hypothesis.

[,] = [,] – [,]

Any particular piece of evidence either supports or denies a hypothesis (but not both), a single number suffices for each rule to define both the MB and MD and thus the CF. CF's reflect assessments of the strength of the evidence in support of the hypothesis.

CF's need to be combined to reflect the operation of multiple pieces of evidence and multiple rules applied to a problem. The combination scenarios are:

1. Several rules all provide evidence that relates to a single hypothesis
2. Our belief is a collection of several propositions taken together
3. The output of one rule provides the input to another



We must first need to describe some properties that we like combining functions to satisfy:

- Combining function should be commutative and Associative
- Until certainty is reached additional conforming evidence should increase MB
- If uncertain inferences are chained together then the result should be less certain than either of the inferences alone

Several rules provide evidences that related to single hypothesis

The measure of belief and disbelief of a hypothesis given two observations s1 and s2 are computed from:

$$MB[h, s_1 \wedge s_2] = \begin{cases} 0 & \text{if } MD[h, s_1 \wedge s_2] = 1 \\ MB[h, s_1] + MB[h, s_2] \cdot (1 - MB[h, s_1]) & \text{otherwise} \end{cases}$$

$$MD[h, s_1 \wedge s_2] = \begin{cases} 0 & \text{if } MB[h, s_1 \wedge s_2] = 1 \\ MD[h, s_1] + MD[h, s_2] \cdot (1 - MD[h, s_1]) & \text{otherwise} \end{cases}$$

One way to state these formulas in English is that

- The measure of belief in h is 0 if h is disbelieved with certainty.
- Otherwise, the measure of belief in h given two observations is the measure of belief given only one observation plus some increment for the second observation.
- This increment is computed by first taking the difference 1 (certainty) and the belief given only the first observation.
- This difference is the most that can be added by the second observation. The difference is then scaled by the belief in h given only the second observation.

From MB and MD, CF can be computed. If several sources of corroborating evidence are pooled, the absolute value of CF will increase. If conflicting evidence is introduced, the absolute value of CF will decrease.

A simple example shows how these functions operate. Suppose we make an initial observation that confirms our belief in h with $MB = 0.3$. Then $MD[h,s_1] = 0$ and $CF[h, s_1] = 0.3$. Now we make a second observation, which also confirms h, with $MB[h,s_2] = 0.2$. Now:

$$\begin{aligned} MB[h,s_1 \wedge s_2] &= 0.3 + 0.2 \cdot 0.7 \\ &= 0.44 \\ MD[h,s_1 \wedge s_2] &= 0.0 \\ CF[h,s_1 \wedge s_2] &= 0.44 \end{aligned}$$

Using MYCIN's rules for inexact reasoning, compute *CF*, *MB*, and *MD* of h_1 given three observations where

$$CF(h_1, o_1) = 0.5$$
$$CF(h_1, o_2) = 0.3$$
$$CF(h_1, o_3) = -0.2$$

$$MB[h_1, o_1] = 0.5$$

$$MB[h_1, o_1 \wedge o_2] = 0.5 + (0.3 \times (1 - 0.5)) = 0.65$$

$$MB[h_1, o_1 \wedge o_2 \wedge o_3] = 0.65$$

$$MD[h_1, o_1] = 0$$

$$MD[h_1, o_1 \wedge o_2] = 0$$

$$MD[h_1, o_1 \wedge o_2 \wedge o_3] = 0.2$$

$$CF[h_1, o_1 \wedge o_2 \wedge o_3] = 0.65 - 0.2 = 0.45.$$

Our belief is a collection of several propositions taken together

We need to compute the certainty factor of a combination of hypothesis. This is necessary when we need to know the certainty factor of a rule antecedent that contains several clauses. The combination certainty factor can be computed from its MB and MD. The formula for the MB of the conjunction {condition of being joined, proposition resulting from the combination of two or more propositions using the ^ operator} and disjunction {proposition resulting from the combination of two or more propositions using the v (OR) operator} of two hypotheses are:

$$[1\ 2,\ ] = \min([1,\ ],\ [2,\ ])$$
$$[1\ 2,\ ] = \max([1,\ ],\ [2,\ ])$$

MD can be computed analogously.

Output of one rule provides the input to another

In this rules are chained together with the result that the uncertain outcome of one rule must provide the input to another. The solution to this problem will also handle the case in which we must assign a measure of uncertainty to initial inputs. This could easily happen in situations where the evidence is the outcome of an experiment or a laboratory test whose results are not completely accurate.

The certainty factor of the hypothesis must take into account both the strength with which the evidence suggests the hypothesis and the level of confidence in the evidence. Let MB'[h,s] be the measure of belief in h given that we are absolutely sure of the validity of s. Let e be the

evidence that led us to believe in s (for example, the actual readings of the laboratory instruments or

results of applying other rules). Then:

$$MB[h, s] = MB'[h, s] \cdot \max(0, CF[s, e])$$

MB which can be thought of as a proportionate decrease in disbelief in h as a result of e as:

$$MB[h, e] = \begin{cases} 1 & \text{if } P(h) = 1 \\ \dfrac{\max[P(h|e), P(h)] - P(h)}{1 - P(h)} & \text{otherwise} \end{cases}$$

MD is the proportionate decrease in belief in h as a result of e

$$MD[h, e] = \begin{cases} 1 & \text{if } P(h) = 0 \\ \dfrac{\min[P(h|e), P(h)] - P(h)}{-P(h)} & \text{otherwise} \end{cases}$$

It turns out that these definitions are incompatible with a Bayesian view of conditional probability. Small changes to them however make them compatible. We can redefine MB as

$$MB[h, e] = \begin{cases} 1 & \text{if } P(h) = 1 \\ \dfrac{\max[P(h|e), P(h)] - P(h)}{(1 - P(h) \cdot P(h|e)} & \text{otherwise} \end{cases}$$

The definition of MD must also be changed similarly.

MYCIN uses CF. The CF can be used to rank hypothesis in order of importance. Example, if a patient has certain symptoms that suggest several possible diseases. Then the disease with higher CF would be investigated first. If E then H → CF(rule) = level of belief of H given E.

Example: CF(E) = CF(it will probably rain today) = 0.6 Positive
CF means evidence supports hypothesis.

MYCIN Formulas for all three combinations:

(i) Make the assumptions that all the rules are independent (ii)
The burden of guarantee independence is on rule writer

(iii) If each combination of scenarios are considered then independent assumption is violated because of large volumes of conditions

The first scenario (a), Our example rule has three antecedents with a single CF rather than three separate rules; this makes the combination rules unnecessary. The rule writer did this because the three antecedents are not independent.

To see how much difference MYCIN's independence assumption can make, suppose for the moment that we had instead had three separate rules and that the CF of each was 0.6. This could happen and still be consistent with the combined CF of 0.7 if three conditions overlap

$$\begin{aligned} MB[h, s \wedge s_2] &= 0.6 + (0.6 \cdot 0.4) \\ &= 0.84 \\ MB[h, (s_1 \wedge s_2) \wedge s_3] &= 0.84 + (0.6 \cdot 0.16) \\ &= 0.936 \end{aligned}$$

substantially. If we apply the MYCIN combination formula to the three separate rules, we get

This is a substantially different result than the true value, as expressed by the expert of 0.7.

Let's consider what happens when independence assumptions are violated in the scenario of (c):

S: sprinkler was on last night
W: grass is wet
R: it rained last night

We can write MYCIN-style rules that describe predictive relationships among these three events:

```
If: the sprinkler was on last night
then there is suggestive evidence (0.9) that
     the grass will be wet this morning
```

Taken alone, this rule may accurately describe the world. But now consider a second rule:

```
If: the grass is wet this morning
then there is suggestive evidence (0.8) that
     it rained last night
```

Taken alone, this rule makes sense when rain is the most common source of water on the grass. But if the two rules are applied together, using MYCIN's rule for chaining, we get

$MB[W,S] = 0.8$ {sprinkler suggests wet}
$MB[R,W] = 0.8 \cdot 0.9 = 0.72$ {wet suggests rains}

BAYESION NETWORKS

CFs is a mechanism for reducing the complexity of a Bayesian reasoning system by making some approximations to the formalism. Bayesian networks in which we preserve the formalism and rely instead on the modularity of the world we are trying to model. Bayesian Network is also called Belief Networks.

The basic idea of Bayesian Network is knowledge in the world is modular. Most events are conditionally independent of other events. Adopt a model that can use local representation to allow interactions between events that only affect each other. The main idea is that to describe the real world it is not necessary to use a huge list of joint probabilities table in which list of probabilities of all conceivable combinations of events. Some events may only be unidirectional others may be bidirectional events may be casual and thus get chained tighter in network.

Implementation:

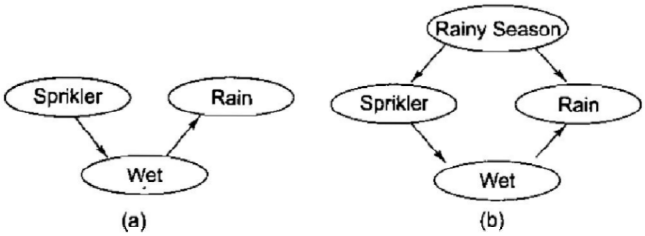
A Bayesian Network is a directed acyclic graph. A graph where the directions are links which indicate dependencies that exist between nodes. Nodes represent propositions about events or events themselves. Conditional probabilities quantify the strength of dependencies.

Eg: Consider the following facts

S: Sprinklers was on the last night

W: Grass is wet

R: It rained last night



From the above diagram, Sprinkler suggests Wet and Wet suggests Rain. (a) shows the flow of constraints.

There are two different ways that propositions can influence the likelihood of each other.

- The first is that causes. Influence the likelihood of their symptoms.
- The second is that the symptoms affect the likelihood of all of its possible causes.

Rules:

(i) If the sprinkler was ON last night then the grass will be wet this morning

(ii) If grass is wet this morning then it rained last night

(iii) By chaining (if two rules are applied together) we believe that it rained because we believe that sprinkler was ON.

The idea behind the Bayesian network structure is to make a clear distinction between these two kinds of influence.

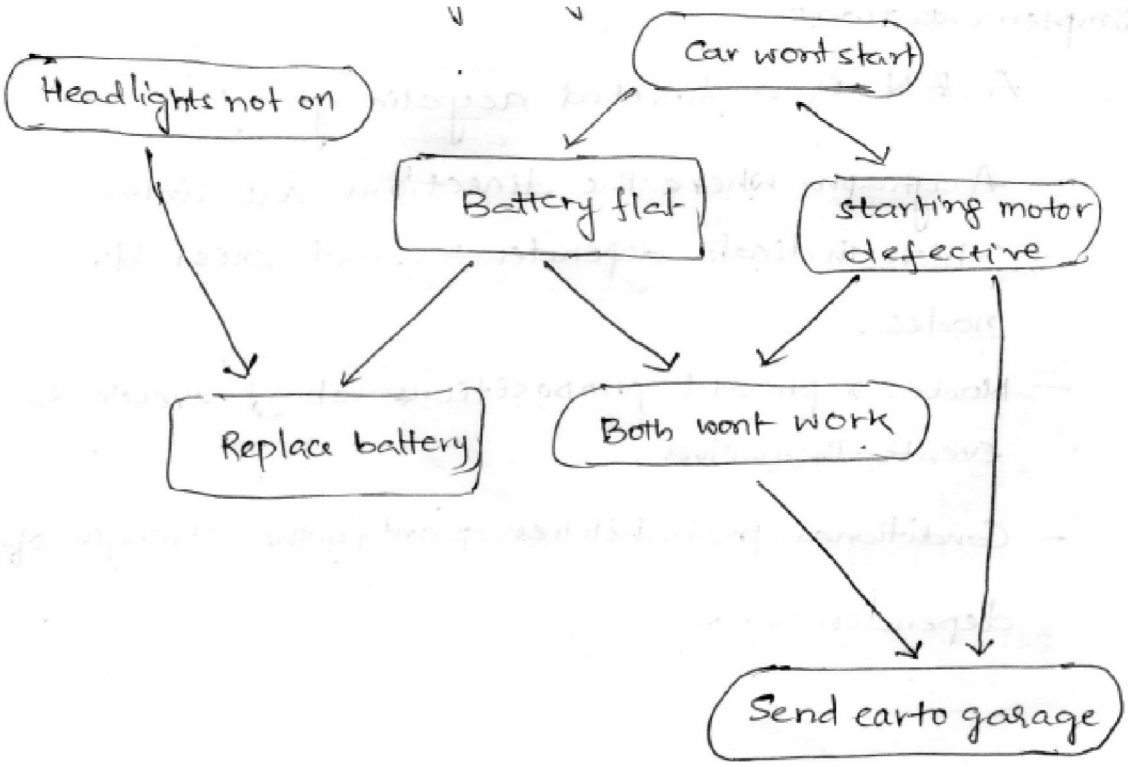
Bayesian Network Example:

If my car won't start then it is likely that

- battery is flat (or)
- starting 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



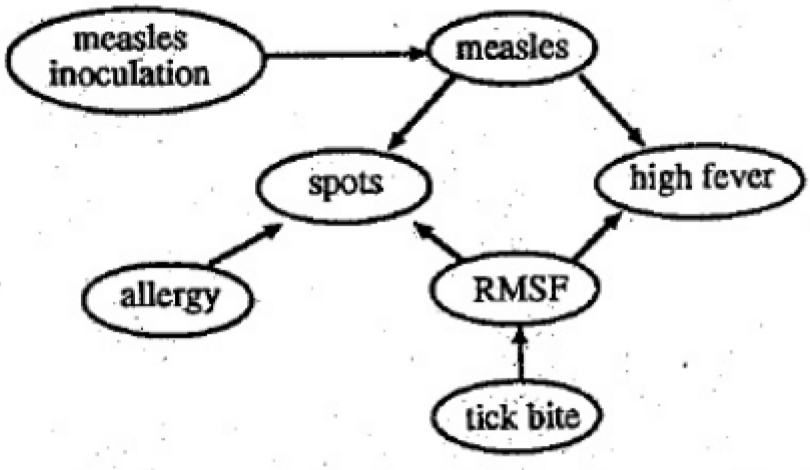
Consider the following set of propositions:

patient has spots
patient has measles
patient has high fever

patient has Rocky Mountain Spotted Fever patient has previously been innoculated against measles patient was recently bitten by a tick

patient has an allergy

Create a network that defines the casual connections among these nodes.



We have four binary elements, represent by four random variables

W = true : Grass is wet

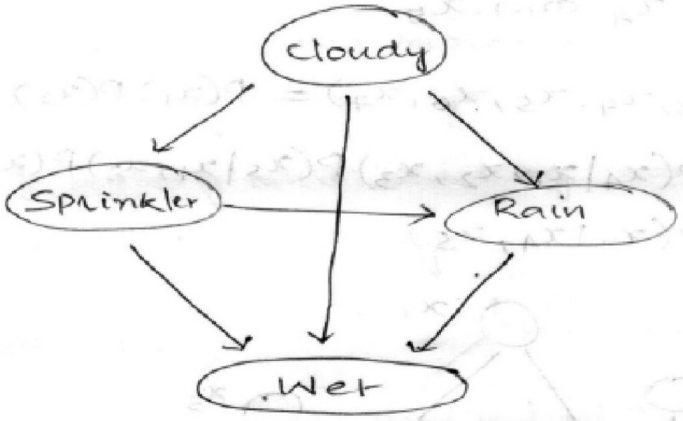
S = true : Water Sprinkler is on

R = true : It is raining

C = true : It is cloudy

By chain rule

$$P(C, S, R, W) = P(C)\, P(S|C)\, P(R|C, S)\, P(W|C, S, R)$$

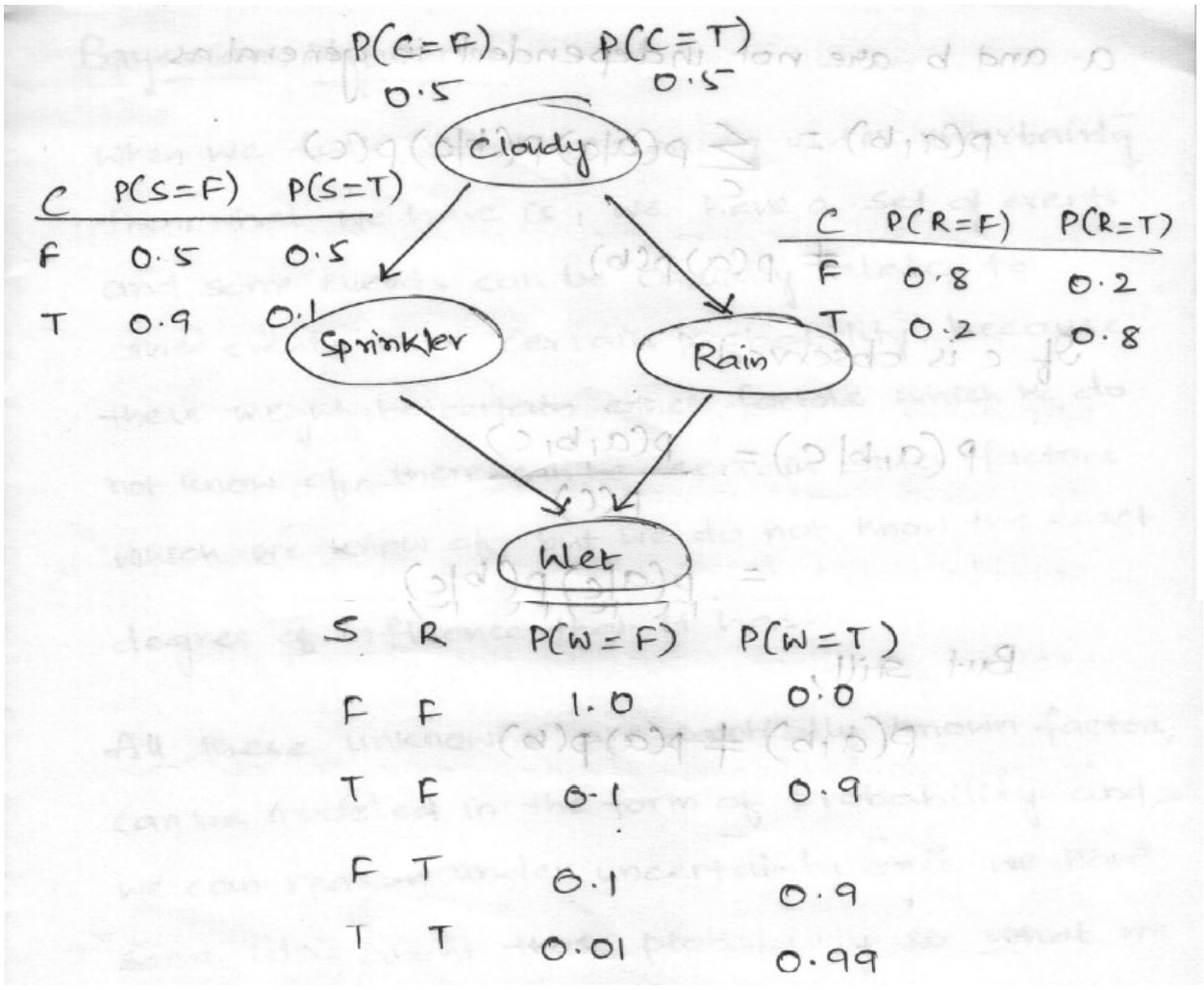


Conditional independence relationships allow us to represent the joint more compactly.

$$P(C, S, R, W) = P(C)\, P(S|C)\, P(R|C)\, P(W|S, R)$$

Attribute	Probability
p(Wet\Sprinkler, Rain)	0.95
P(Wet\Sprinkler, ¬Rain)	0.9
p(Wet\¬Sprinkler, Rain)	0.8
p(Wet\¬Sprinkler, ¬Rain)	0.1
p(Sprinkler\RainySeason)	0.0
p(Sprinkler\¬RainySeason)	1.0
p(Rain \RainySeason)	0.9
p(Rain \ ¬RainySeason)	0.1
p(RainySeason)	0.5

Conditional Probabilities for a Bayesian Network



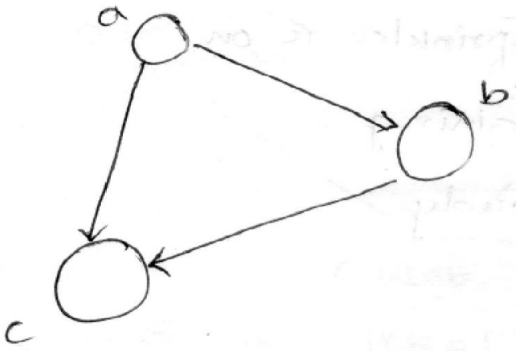
Conditional Probability Table

Each node in Bayesian Network has an associated Conditional Probability table (CPT). This gives the probability values for the random variable at the node conditioned on values for its parents.

$$P(W = T|S = T, R = F) = 0.9 \Rightarrow P(W = F|S = T, R = F) = 1 - 0.9 = 0.1$$

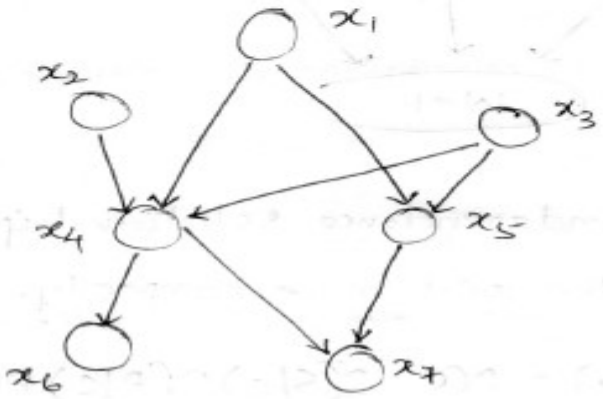
Since each row must sum to one. Since the C node has no parents, its CPT specifies the prior probability that is cloudy (in this case, 0.5).

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



Conditional Independence: x_7 is only dependent on x_4 and x_5.

$$P(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = P(x_1)\,P(x_2)\,P(x_3)$$
$$P(x_4|x_1, x_2, x_3)\,P(x_5|x_1, x_3)\,P(x_6|x_4)$$
$$P(x_7|x_4, x_5)$$



Dempster-Shafer Theory

So far we considered individual propositions and assign each of them a point of degree of belief that is warranted for given evidence. The Dempster Shafer theory approach considers sets of propositions and assigns each of them an interval

{ , }

in which the degree of belief must lie.

Belief measures the strength of evidence in favor of the set of propositions. It ranges from 0 to 1 where 0 indicates no evidence and 1 denoting certainty.

Plausability (PL) is defined as

( ) = 1 − ( )

It also ranges from 0 to 1 and measures the extent to which evidence in favour of ¯S leaves room for belief in S.

The confidence interval is then defined as [*B*(*E*),*PL*(*E*)]
where

$$B(E) = \sum_{A \subseteq E} M$$

where *i.e.* all the evidence that makes us believe in the correctness of *P*, and

$$PL(E) = 1 - B(\neg E) = 1 - \sum_{\neg A \subseteq \neg E} M$$

where *i.e.* all the evidence that contradicts *P*.

Set up a confidence interval – an interval of probabilities within which the true probability lies with a certain confidence based on belief B and plausibility PL provided by some evidence E for a proposition P.

Suppose we are given two belief statements M1 ^ M2. Let S be the subset of Θ which M1 assigns a

non-zero value & let y be corresponding set to M2. We define the combination M3 of M1 & M2.

$$m_3(Z) = \frac{\sum_{X \cap Y = Z} m_1(X) \cdot m_2(Y)}{1 - \sum_{X \cap Y = \phi} m_1(X) \cdot m_2(Y)}$$

$fever = \{flu, cold, pneu\}$
$\Theta = 1.0$ (total probability)

$\{Flu, Cold, Pneu\}$	(0.6)	m1
$\{\Theta\}$	(0.4)	

Suppose m_2 corresponds to our belief after observing a runny nose:

$\{All, Flu, Cold\}$	(0.8)
Θ	(0.2)

Consider the same propositions again, and assume our task is to identify the patient's disease using Dempster-Shafer theory.

(a) What is Θ?

(b) Define a set of m functions that describe the dependencies among sources of evidence and elements of Θ.

(c) Suppose we have observed spots, fever, and a tick bite. In that case, what is our $Bel(\{RockyMountainSpottedFever\})$?

- $\Theta = \{M, A, R\}$:

  M: measles
  A: allergy
  R: RMSF

- In all of these, remember that we are assuming that Θ contains the universe of possibilities, so our patient must have one of the three diseases. Also, notice that these numbers aren't the same as the numbers in problem 4. There the numbers described probabilities of symptoms given diseases. Here we're talking about likelihoods of diseases given symptoms.

m_1 (belief given just spots):

$\{M, A, R\}$	(0.9)
$\{\Theta\}$	(0.1)

m_2 (belief given just fever):

$\{M, R\}$	(0.9)
$\{\Theta\}$	(0.1)

m_3 (belief given just measles shot):

$\{A, R\}$	(0.95)
$\{\Theta\}$	(0.05)

m_4 (belief given just tick bit):

$\{R\}$	(0.95)
$\{\Theta\}$	(0.05)

Combining m_1 and m_2 to form m_5 (belief given spots and fever):

		$\{M, R\}$	(0.9)	Θ	(0.1)
$\{M, A, R\}$	(0.9)	$\{M, R\}$	(0.81)	$\{M, A, R\}$	(0.09)
Θ	(0.1)	$\{M, R\}$	(0.09)	Θ	(0.01)

The total $Bel(\{M, R\})$ at this point is .9.

Combining m_5 and m_4 (belief given spots, fever, and tick bite):

		$\{R\}$	(0.95)	Θ	(0.05)
$\{M, R\}$	(0.9)	$\{R\}$	(0.855)	$\{M, R\}$	(0.045)
$\{M, A, R\}$	(0.09)	$\{R\}$	(0.0855)	$\{M, A, R\}$	(0.0045)
Θ	(0.01)	$\{R\}$	(0.0095)	Θ	(0.0005)

So our overall $Bel(\{R\})$ is 0.95.

Fuzzy Logic

Fuzzy logic is an alternative for representing some kinds of uncertain knowledge. Fuzzy logic is a form of many-valued logic; it deals with reasoning that is approximate rather than fixed and exact. Compared to traditional binary sets (where variables may take on true or false values), fuzzy logic variables may have a truth value that ranges in degree between 0 and 1. Fuzzy logic has been extended to handle the concept of partial truth, where the truth value may range between completely true and completely false. Fuzzy set theory defines set membership as a possibility distribution.

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*. The general rule for this can expressed as:

  $$f : [0,1]^n \rightarrow [0,1].$$

  where *n* some number of possibilities.

  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 sets *etc.* and reason effectively.

Intersection Search

8. Weak Slot and Filler Structures

Weak slot and filler structures turns out to be useful one for reasons besides the support of inheritance, though, including

Introduction

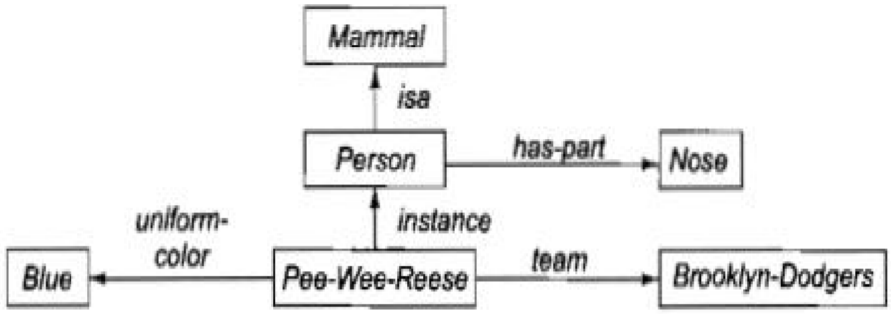
- It enables attribute values to be retrieved quickly assertions are indexed by the entities
  - binary predicates are indexed by first argument.
  - *E.g. team(Mike-Hall , Cardiff)* .
- Properties of relations are easy to describe.
- It allows ease of consideration as it embraces aspects of object oriented programming including modularity and ease of viewing by people.

Weak slot and filler structures describe two views: **semantic nets and frames**. These talk about the representations themselves and about techniques for reasoning with them. They do not say much about the specific knowledge that the structures should contain. We call these as "knowledge poor" structures.

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.

Semantic Nets

Semantic Nets were originally designed as a way to represent the meaning of English words. The main idea is that the meaning of a concept comes from the ways in which it is connected to other concepts. The information is stored by interconnecting nodes with labeled arcs. Semantic nets
initially we used to represent labeled connections between nodes.



A Semantic Network

These values can also be represented in logic as: *isa(person, mammal), instance(Pee-Wee-Reese, person) team(Pee-Wee-Reese, Brooklyn-Dodgers).* This network contains examples of both the isa and instance relations as well as some other more domain-specific relations. In this network, we could use inheritance to derive the additional relation
has-part(Pee-Wee-Reese,Nose)

The Semantic Nets can be represented in different ways by using relationships. Semantic nets have been used to represent a variety of knowledge in a variety of different programs.
One of the early ways that semantic nets were used was to find relationships among objects by

spreading activation out from each of two nodes and seeing where the activation met. This process is called ***"intersection search"***.

Using this process, it is possible to use the network to answer questions such as ***"What is the connection between the Brooklyn Dodgers and blue?"***

Representing Non-binary predicates

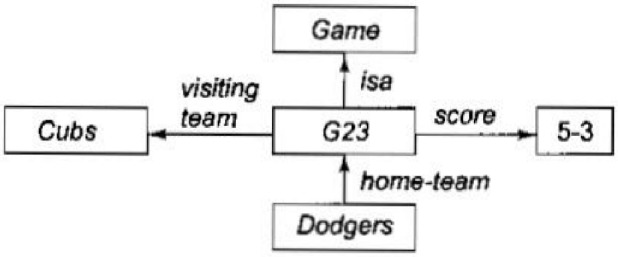
Semantic nets are natural way to represent relationships that would appear as ground instances of binary predicates in predicate logic. Arcs from the figure could be represented in logic as

- *isa(Person, Mammal)*
- *instance(Pee-Wee-Reese, Person)*
- *team(Pee-Wee-Reese, Brooklyn-Dodgers)*
- *uniform-color(Pee-Wee-Reese, Blue)*

Many unary predicates in logic can be thought as binary predicates using isa and instance.

Example: man(marcus) → instance(Marcus,man)

Three or more place predicates can also be converted to a binary form by creating one new object representing the entire predicate statement and then introducing binary predicates to describe the relationship to this new object of each of the original arguments. ***Example:*** score(Cubs, Dodgers, 5-3) can be represented in semantic net by creating a node to represent the specific game & then relating each of the three pieces of information to it.

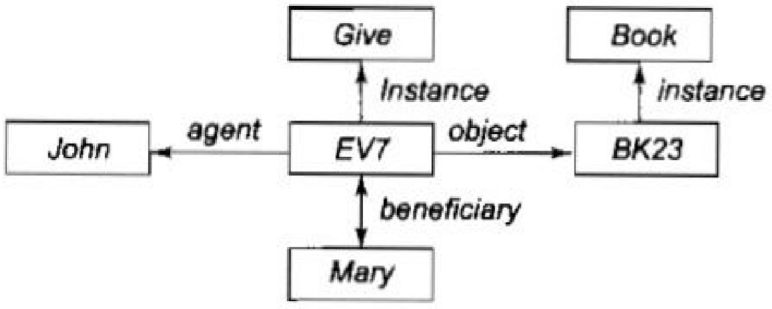


A Semantic Net for an n-Place Predicate

This technique is particularly useful for representing the contents of a typical declarative sentence

that describes several aspects of a particular event.

Example: John gave the book to Mary.



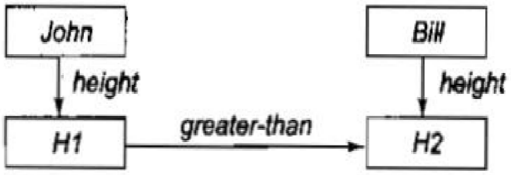
Making Some Important Distinctions

In the networks, some distinctions are glossed that are important in reasoning. For example, there should be difference between a link that defines a new entity and one that relates two existing entities.

John height is 72

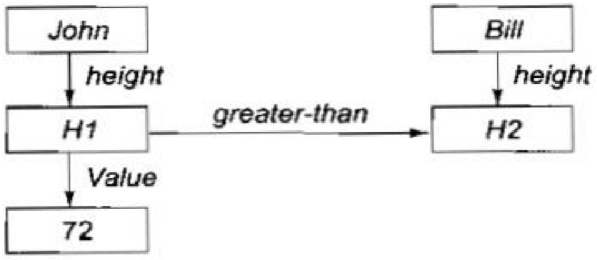
Both nodes represent objects that exist independently of their relationship to each other.

John is taller than Bill.



H1 and H2 are new concepts representing John's height and Bill's height. They are defined by their relationships to the nodes John and Bill.

John is taller than Bill and his height is 72.



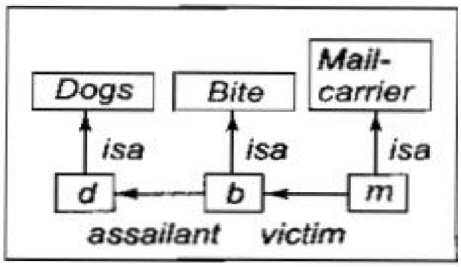
The procedures that operate on nets such as this can exploit the fact that some arcs such as define new entities, while others such as ***greater-than*** and ***value***, merely describe relationships among existing entities.

height

Partitioned Semantic Nets

To represent simple quantified expressions in semantic nets. The way is to partition the semantic net into a hierarchical set of spaces, each of which corresponds to the scope of one or more variables.

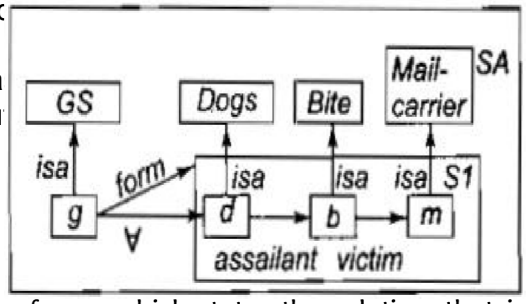
The statement: ***The dog bit the mail carrier.***



The statement: ***Every dog has bitten a mail-carrier***

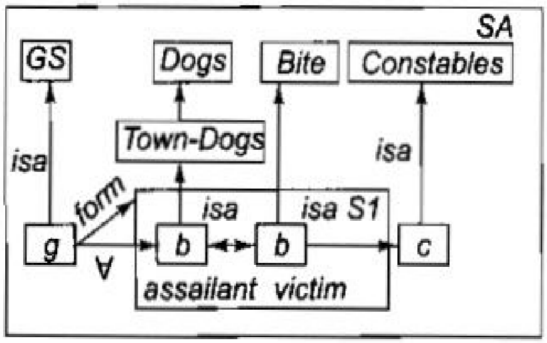
$\forall x \text{Dog}(x) \rightarrow \exists y \text{Mail-Carrier}(y) \wedge \text{Bite}(x,y)$

The nodes Dogs, Bite, Mail-carrier represent the classes of dogs, biting and mail carriers respectively, while the nod…og, biting and a particular mailcarrier. This fact can be …ut no partitioning.
The node g stands for the a…ance of the special class GS of general statements abou…antifiers). Every element of GS



has at least two attributes: a form, which states the relation that is being asserted, and one or more ∀ connections, one for each of the universally quantified variables. In this example, for *every dog d, there exists a biting event b, and a mail-carrier m, such that d is the assailant of b and m is the victim.*

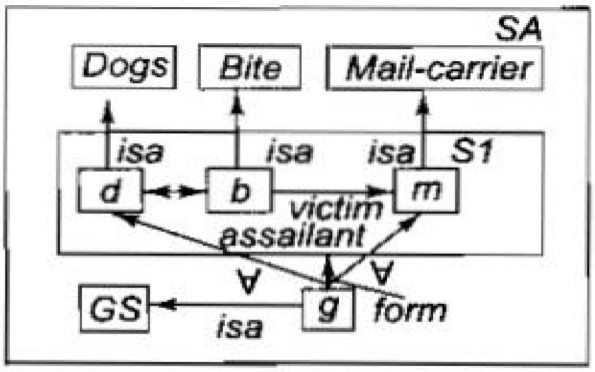
The statement: ***Every dog in town has bitten the constable***



In this net, the node c representing the victim lies outside the form of the general statement. Thus it is not viewed as an existentially quantified variable whose value may depend on the value of d.

T

Every dog has bitten every mail carrier.



In this case, g has two ∀ links, one pointing to d, which represents any dog and one pointing to m, representing any mail carrier.
he statement.

FRAMES

Evolution into Frames

-
-
-

Frames

-

As we expand the range of problem solving tasks that the representation must support, the representation necessarily begins to become more complex.

It becomes useful to assign more structure to nodes as well as to links.

The more structure the system has, the more likely it is to be termed a frame system.

Natural language understanding requires inference i.e.,

- assumptions about what is typically true of the objects or
- Situations under consideration.

- Such information can be coded in structures known as 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.
- Frame is a type of schema used in many AI applications including vision and natural language processing.
- Frames provide a convenient structure for representing objects that are typical to stereotypical situations. The situations to represent may be visual scenes, structure of complex physical objects, etc.
- Frames are also useful for representing commonsense knowledge. As frames allow nodes to have structures they can be regarded as three-dimensional representations of knowledge.
- A frame is similar to a record structure and corresponding to the fields and values are slots and slot fillers. Basically it is a group of slots and fillers that defines a stereotypical object.
- A single frame is not much useful.
- Frame systems usually have collection of frames connected to each other. Value of
- an attribute of one frame may be another frame. A frame for a book is given below.

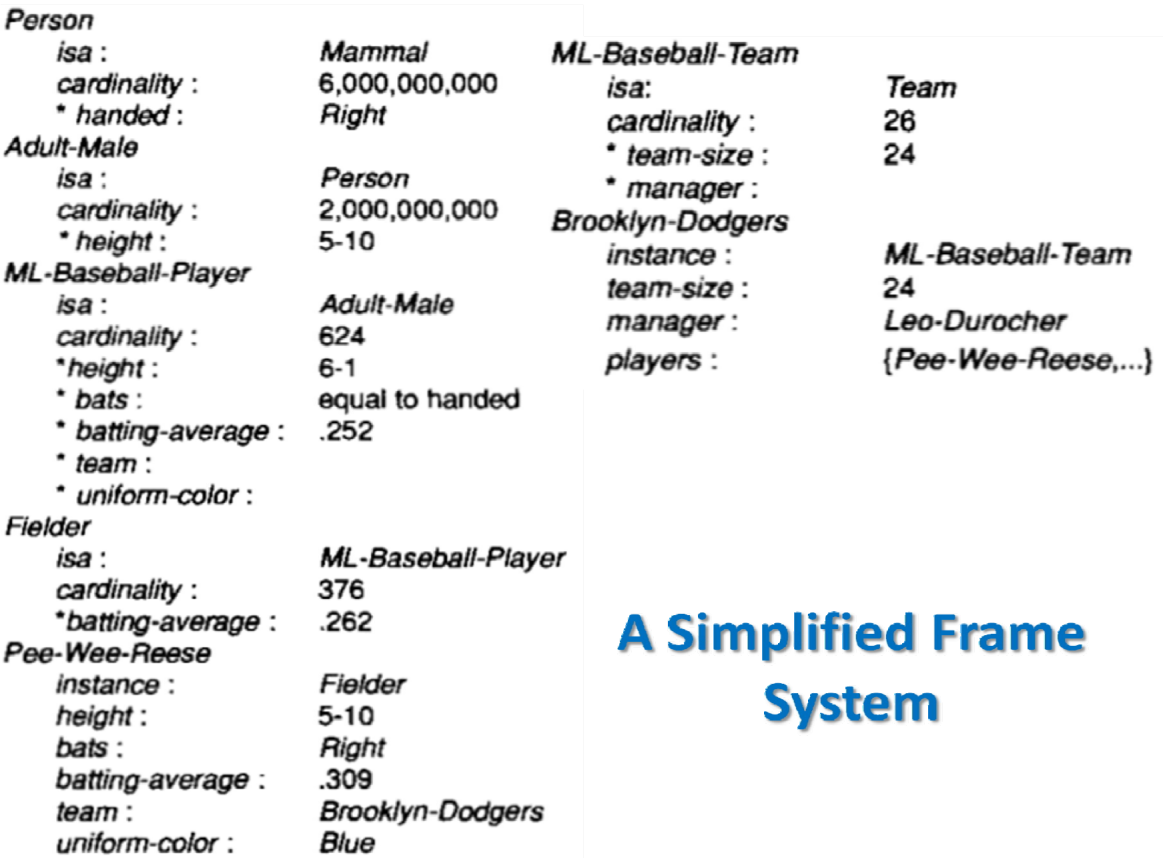
Slots	Fillers
publisher	Thomson
title	Expert Systems
author	Giarratano
edition	Third
year	1998
pages	600

Frames as Sets and Instances

- Set theory provides a good basis for understanding frame systems.
-

Each frame represents:

- a class (set), or
- an instance (an element of a class).



- The is-a relationship is in fact the subset relation.
  - The set of adult males is a subset of the set of people.
  - The set of major league baseball players is a subset of adult males and so forth
- Theinstance relation corresponds to the relation element-of.
  - Wee-Reese is the element of Fielder.

Pee-

A

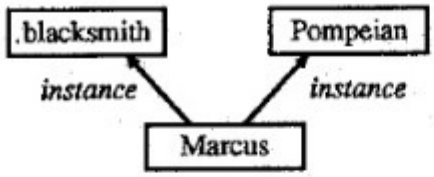
- class represents a set.
- There are 2 kinds of attributes that can be associated with it:
  - Attributes about the set itself and
  - Attributes that are to be inherited by each element of the set
    - Prefixed with *

Weak Slot Filler Structures (Continued)

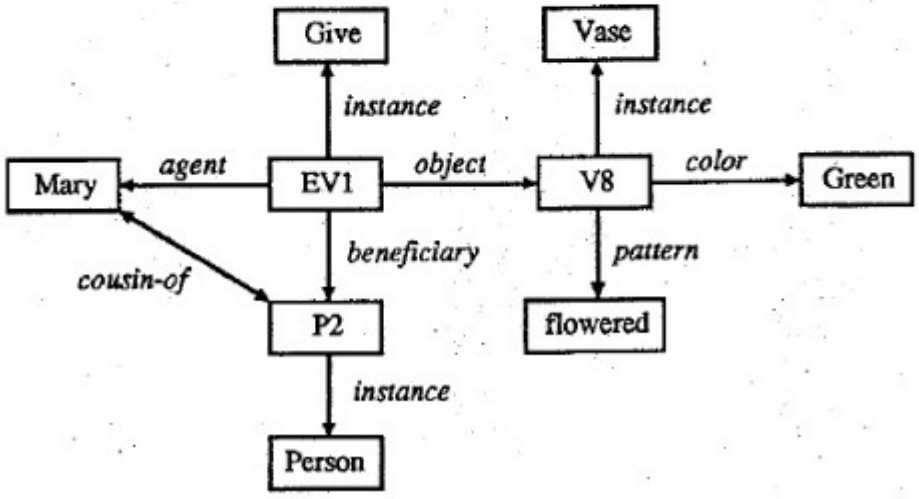
- *Slot and Filler Structures* are a device to support property inheritance along isa and instance links.
  - Knowledge in these is structured as a set of entities and their attributes.
- This structure turns out to be useful for following reasons:
  - It enables attribute values to be retrieved quickly
    - assertions are indexed by the entities
    - 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.
    - Modularity
    - Ease of viewing by people.

1. Construct semantic net representations for the following:
   (a) *Pompeian(Marcus), Blacksmith(Marcus)*
   (b) Mary gave the green flowered vase to her favorite cousin.

(a) Marcus:



(b) Mary:



Inheritance

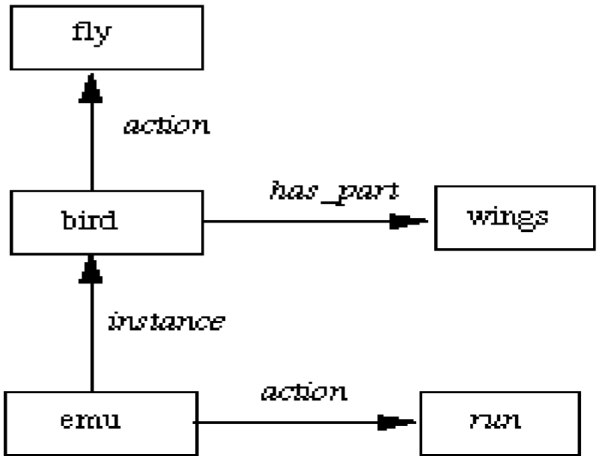
-- the *isa* and *instance* representation provide a mechanism to implement this.

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:



A Semantic Network for a Default Reasoning

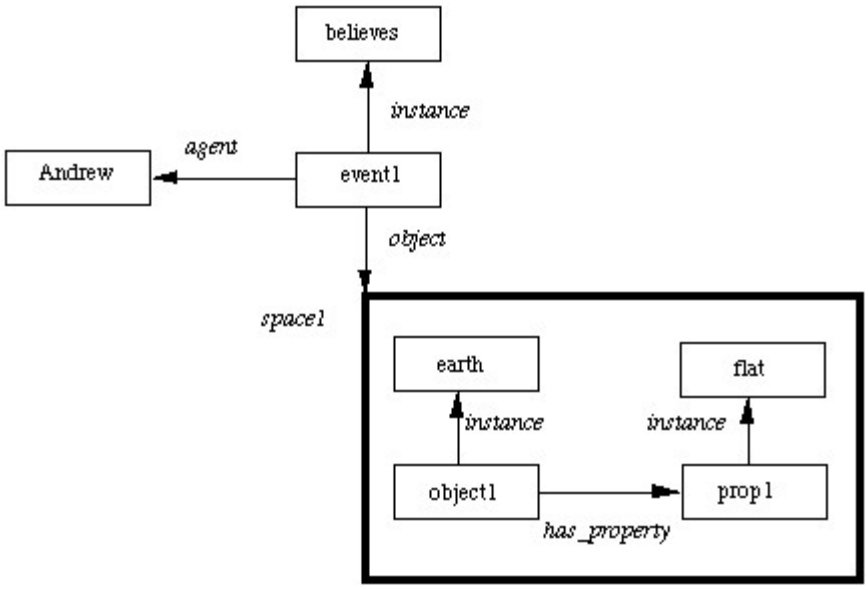
Partitioned Networks

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. We can the have nodes and arcs to link this *space* the the rest of the network to represent Andrew's belief.



WHAT ARE FRAMES?

Natural language understanding requires inference i.e., assumptions about what is typically true of the objects or situations under consideration. Such information can be coded in structures known as frames.

NEED OF FRAMES

Frame is a type of schema used in many AI applications including vision and natural language processing. Frames provide a convenient structure for representing objects that are typical to stereotypical situations. The situations to represent may be visual scenes, structure of complex physical objects, etc. Frames are also useful for representing commonsense knowledge. As frames allow nodes to have structures they can be regarded as three-dimensional representations of knowledge.

A frame is similar to a record structure and corresponding to the fields and values are slots and slot fillers. Basically it is a group of slots and fillers that defines a stereotypical object. A single frame is not much useful. Frame systems usually have collection of frames connected to each other. Value of an attribute of one frame may be another frame. A frame for a book is given below.

Slots	Fillers
publisher	Thomson
title	Expert Systems
author	Giarratano

edition	Third
year	1998
pages	600

The above example is simple one but most of the frames are complex. Moreover with filler slots and inheritance provided by frames powerful knowledge representation systems can be built.

Frames can represent either generic or frame. Following is the example for generic frame.

Slot	Fillers
name	computer
specialization_of	a_kind_of machine
types	(desktop, laptop,mainframe,super) if-added: Procedure ADD_COMPUTER
speed	default: faster if-needed: Procedure FIND_SPEED
location	(home,office,mobile)
under_warranty	(yes, no)

The fillers may values such as computer in the name slot or a range of values as in type's slot. The procedures attached to the slots are called procedural attachments. There are mainly three types of procedural attachments: if-needed, default and if-added. As the name implies if-needed types of procedures will be executed when a filler value is needed. Default value is taken if no other value exists. Defaults are used to represent *commonsense knowledge*. Commonsense is generally used when no more situation specific knowledge is available.

The if-added type is required if any value is to be added to a slot. In the above example, if a new type of computer is invented ADD_COMPUTER procedure should be executed to add that information. An if-removed type is used to remove a value from the slot.

Frame Knowledge Representation

Consider the example first discussed in Semantics Nets :

Person

isa: Mammal Cardinality:

Adult-Male

isa: Person

Cardinality:

Rugby-Player

isa: Adult-Male Cardinality:

Height:
Weight:
Position:
Team:
Team-Colours:

Back
isa: Rugby-Player Cardinality:
Tries:

Mike-Hall
instance: Back Height: 6-0
Position: Centre
Team: Cardiff-RFC
Team-Colours: Black/Blue

Rugby-Team
isa: Team Cardinality: Team-size: 15
Coach:

Cardiff-RFC Rugby-Team
instance: 15
Team-size: Terry Holmes
Coach: {Robert-Howley, Gwyn-Jones, ... }
Players:

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.

Note

- 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

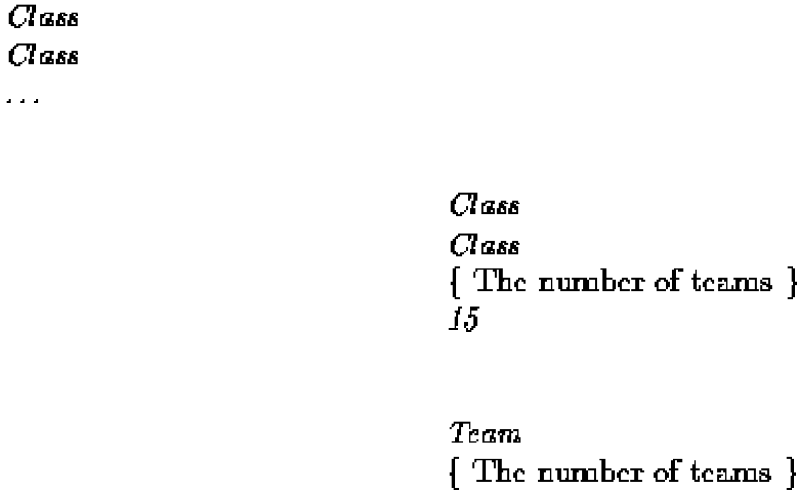
A metaclass is a special class whose elements are themselves classes.

Now consider our rugby teams as:

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.



Rugby-Team
15
Terry Holmes
Class *Back*
instance: *6-0*
isa: *Scrum Half*
Cardinality: *Cardiff-RFC*
Black/Blue
Team
instance:
isa:
Cardinality:
Team-Size:

Rugby-Team
isa:
Cardinality:
Team-size: 15
Coach:

Cardiff-RFC
instance:
Team-size:
Coach:

Robert-Howley
instance:
Height:
Position:
Team:
Team-Colours:

Figure: A Metaclass frame system

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.

Consider the following:

SLOT

isa: Class
instance: Class
domain:
range:
range-constraint:
definition:
default:
to-compute:
single-valued:

Coach

instance: SLOT
domain: Rugby-Team
range: Person
range-constraint: (experience x.manager)
default:
single-valued: TRUE

Colour

instance: SLOT
domain: Physical-Object
range: Colour-Set
single-valued: FALSE

Team-Colours

instance: SLOT
isa: Colour
domain: team-player
range: Colour-Set
range-constraint: not Pink
single-valued: FALSE

Position

instance: SLOT
domain: Rugby-Player
range: { Back, Forward, Reserve }
to-compute: x.position single-valued: TRUE

NOTE the following:

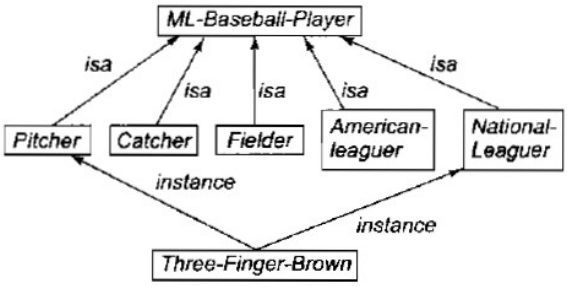
- Instances of *SLOT* are slots
- Associated with *SLOT* are attributes that each instance will inherit.

- 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.

Slots as Full-Fledged Objects

So far, we have provided a way to describe sets of objects and individual objects, both in terms of attributes and values. Thus we have made extensive use of attributes, which we have represented as slots attached to frames. But it turns out that there are several reasons why we would like to be able to represent attributes explicitly and describe their properties. Some of the properties we would like to be able to represent and use in reasoning include:

- The classes to which the attribute can be attached, i.e. for what classes does it make sense? For example, weight makes sense for physical objects but not for conceptual ones (except in some metaphorical sense).
- Constraints on either the type or the value of the attribute. For example, the age of a person must be a numeric quantity measured in some time frame, and it must be less than the ages of the person's biological parents.
- A value that all instances of a class must have by the definition of the class.
- A default value for the attribute.
- Rules for inheriting values for the attribute. The usual rule is to inherit down *isa* and *instance* links. But some attributes inherit in other ways. For example, *last-name* inherits down the *child-of* link.
- Rules for computing a value separately from inheritance. One extreme form of such a rule is a procedure written in some procedural programming language such as LISP.
- An inverse attribute.
- Whether the slot is single-valued or multivalued.



ML-Baseball-Player
is-covered-by : {*Pitcher, Catcher, Fielder*}
{*American-Leaguer, National-Leaguer*}

Pitcher
isa : *ML-Baseball-Player*
mutually-disjoint-with : {*Catcher, Fielder*}

Catcher
isa : *ML-Baseball-Player*
mutually-disjoint-with: {*Pitcher, Fielder*}

Fielder
isa : *ML-Baseball-Player*
mutually-disjoint-with : {*Pitcher, Catcher*}

American-Leaguer
isa : *ML-Baseball-Player*
mutually-disjoint-with : {*National-Leaguer*}

National-Leaguer
isa : *ML-Baseball-Player*
mutually-disjoint-with : {*American-Leaguer*}

Three-Finger-Brown
instance : *Pitcher*
instance : *National-Leaguer*