ECS-801: Artificial Intelligence (AI)

[UNIT -3]

Knowledge Representation & Reasoning: Propositional logic, Theory of first order logic, Inference in First order logic, Forward & Backward chaining, Resolution, Probabilistic reasoning, Utility theory, Hidden Markov Models (HMM), Bayesian Networks.

INFERENCE AND CONTROL

Inference engine performs 2 major tasks:

1) examines existing facts and rules and adds new facts when possible

2) decides the order in which inferences are made.

We shall look at Inference and Control

INFERENCE:

Infer means " to derive as a conclusion from facts or premises".
There are 2 common rules for deriving new facts from rules and known facts.These are Modus Ponens and Modus Tollens

MODUS PONENS

*most common inference strategy
*simple ,reaoning based on it is easily understood.

The rule states that when A is known to be true and if a rule states " If A then B " it is valid to conclude that B is true.

MODUS TOLLENS

When B is false rule If A, then B
then A is false.

E.g:
Rule : IF Ahmet's CAR IS DIRTY
THEN Ahmet HAS BEEN DRIVING OUTSIDE ANKARA

Given fact : Ahmet has not been outside Ankara.
New rule : Ahmet car is not dirty.

This conclusion seems quite obvious but cannot be reached by most expert systems. Because they use modus ponens for deriving new facts from rules.

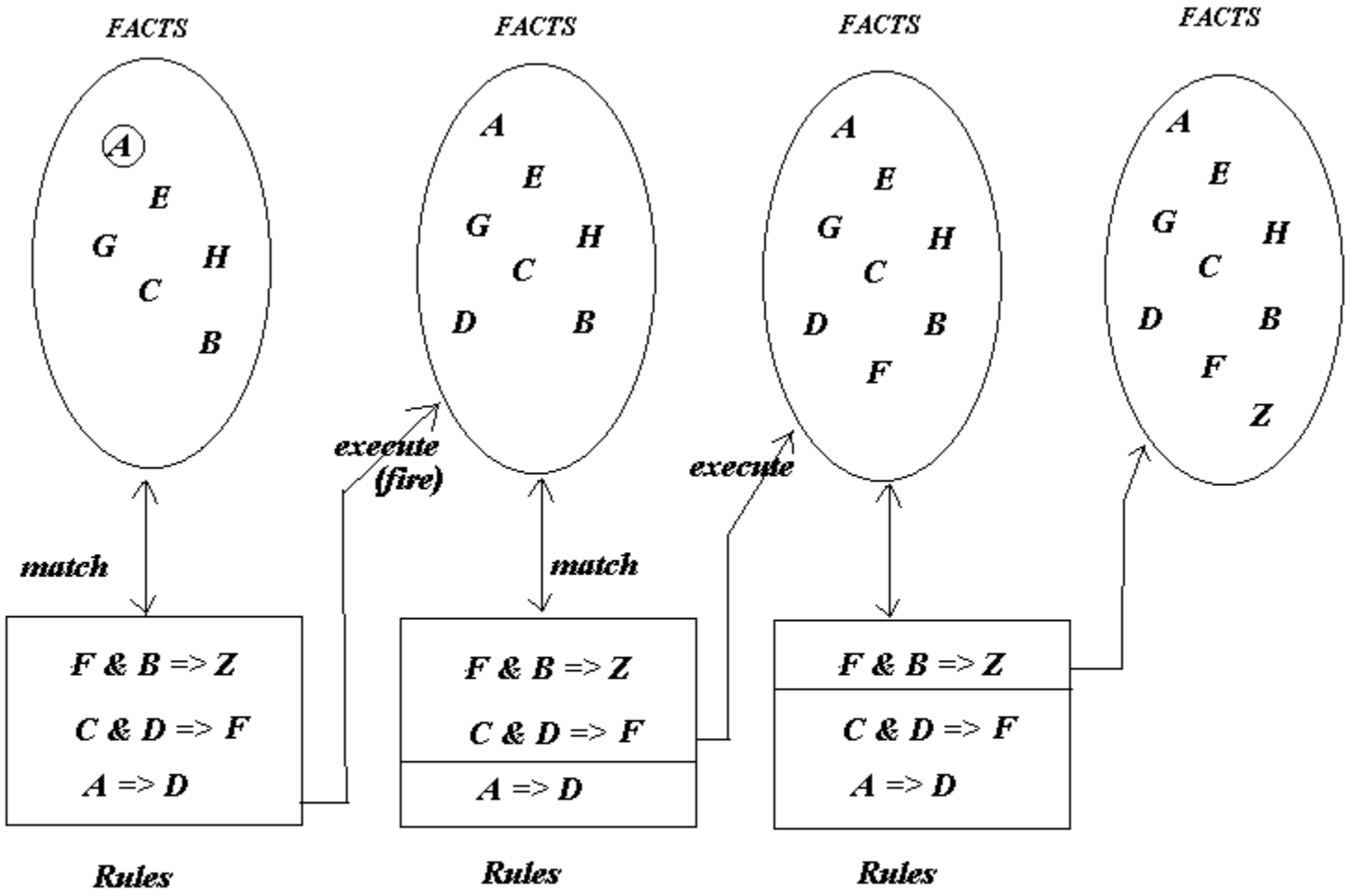
REASONING ABOUT UNCERTAINTY

An inference engine should be able to handle incomplete information. The degree of certainity is represented as a number of attached to a fact (certainty factor).

There are three inferencing methods. These are Forward,Backward and Mixed Chaining.

FORWARD CHAINING:

Forward Chaining(Example 1)



Problem: Does situation Z exists or not ?

The first rule that fires is A->D because A is already in the database. Next we infer D. Existence of C and D causes second rule to fire and as a consequence F is inferred and placed in the database. This in turn, causes

the third rule F?B->Z to fire, placing Z in the database.
This technique is called forward chaining.

A very simple Forward chaining Algorithm

Given m facts $F_1,F_2,....,F_m$? N RULES
$R_1,R_2,.....R_n$

repeat
for i ?- 1 to n do
if one or more current facts match the antecedent of Ri then
1) add the new fact(s) define by the consequent
2) flag the rule that has been fired
3) increase m
until no new facts have been produced.

Forward Chaining (Example 2)

Rule 1
IF the car overheats , THEN the car will stall.
Rule 2
IF the car stalls
THEN it will cost me money
AND I will be late getting home

Now, the question is
How do you arrive at conclusion that this situation will cost money and cause you to be late ?
The condition that triggers the chain of events is the car overheating

BACKWARD CHAINING:

Backward Chaining (Example 1)

Rule 1
IF the car is not tuned AND the battery is weak
THEN not enough current will reach the starter.
Rule 2
IF not enough current reaches the starter
THEN the car will not start.

Given facts:
The car is not tuned
The battery is weak.
Now, the question is

How would you arrive at the conditions that have resulted in the car failing to start?

Backward Chaining(Example 2)

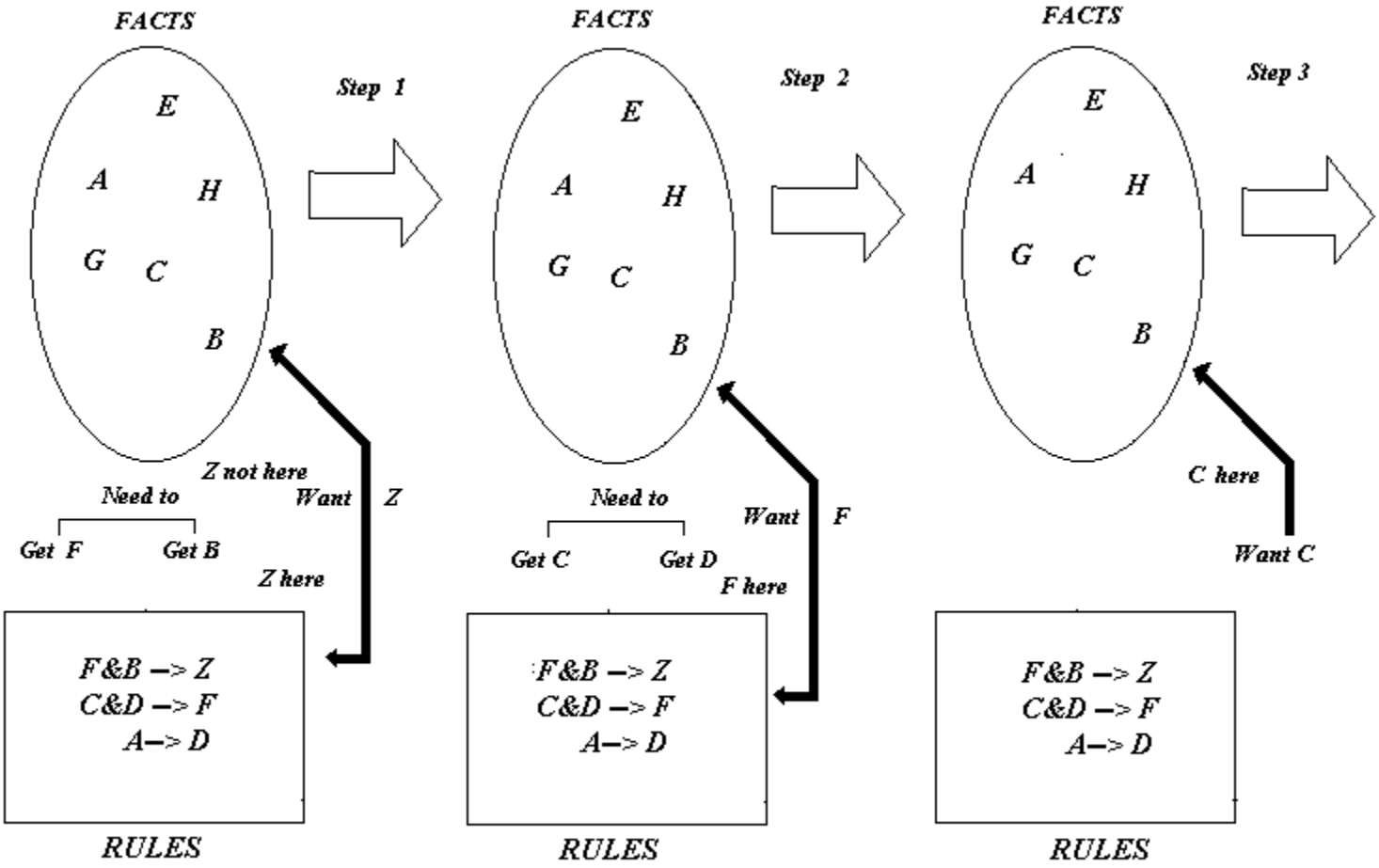
In such a situation backward chaining might be more cost-effective.With this inference method the system starts with what it wants to prove, e.g.,that situation Z exists, and only executes rules that are relavent to establishing it.Figure following shows how bacward chaining would work using the rules from the forward chaining example.

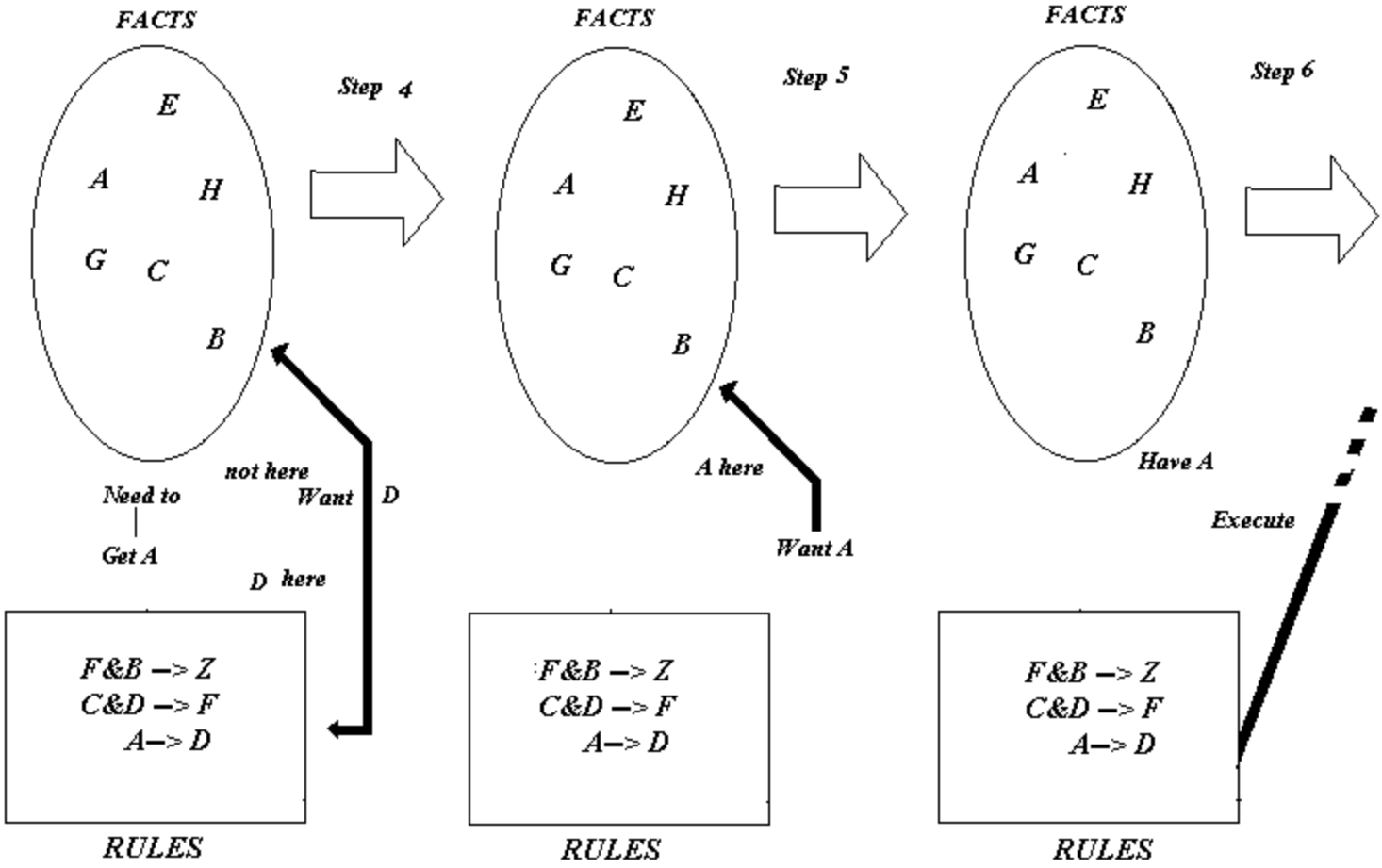
In step 1 the system is told to establish (if it can) that situation Z exists,It first checks the data base for Z,and when that fails, searches for rules that conclude Z,i.e., have Z on the right side of the arrow.It finds the rule F?B->Z, and decides that it must establish F and B in order to conclude Z.

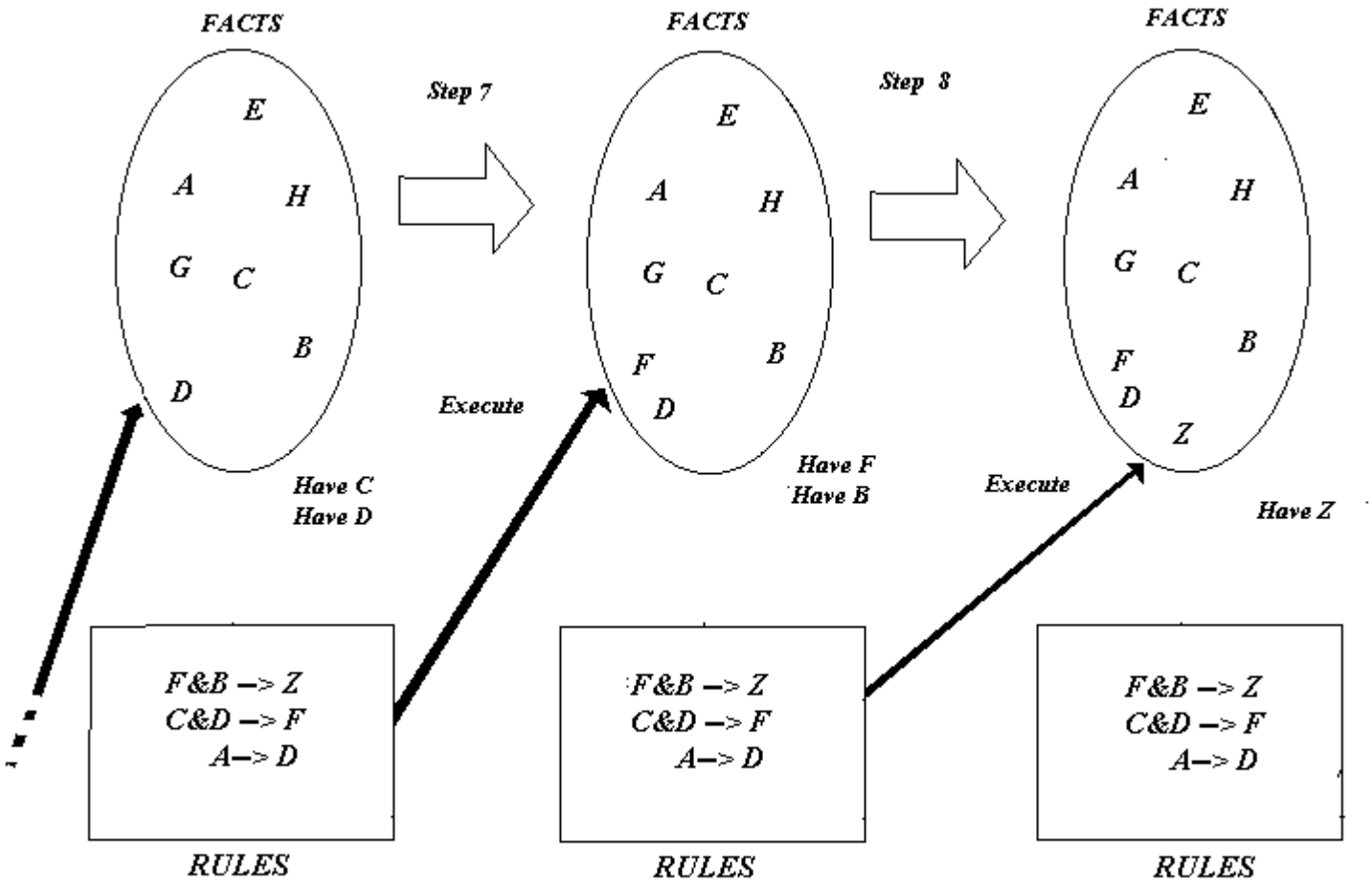
In step 2 the system tries to establish F, first checking the data base and then finding a rule that concludes F. From this rule, C?D->F, the system decides it must establish C and D to conclude F.

In steps 3 through 5 the system finds C in the data base but decides it must establish A before it can conclude D. It then finds A in the data base.

In steps 6 through 8 the system executes the third rule to establish D, then executes the second rule to establish the original goal, Z.The infenece chain created here is identical to the one created by forward chaining. The difference in two approaches hinges on the method in which data and rules are searched.


FACTS
E
A
H
G
C
B
Step 1
Z not here
Need to
Want Z
Get F
Get B
Z here
F&B –> Z
C&D –> F
A–> D
RULES
FACTS
E
A
H
G
C
B
Step 2
Need to
Want F
Get C
Get D
F here
F&B –> Z
C&D –> F
A–> D
RULES
FACTS
E
A
H
G
C
B
Step 3
C here
Want C
F&B –> Z
C&D –> F
A–> D
RULES


FACTS
E
A
H
G
C
B
Step 4
not here
Need to
Want
D
Get A
D here
F&B –> Z
C&D –> F
A–> D
RULES
FACTS
E
A
H
G
C
B
Step 5
A here
Want A
F&B –> Z
C&D –> F
A–> D
RULES
FACTS
E
A
H
G
C
B
Step 6
Have A
Execute
F&B –> Z
C&D –> F
A–> D
RULES



7.5 MIXED CHAINING

Mixed Chaining (Example)

R1. IF F and H then K } Suppose R_1-R_3 are
R2. IF E and A then K } backward chaining
R3. IF E and B then H }

R4. IF A and G then B } R_4-R_8
R5. IF B and D then H } are forward chaining
R6. IF G and D then E }
R7. IF A and B then D }

***R*$_8$.** IF A and C then G }

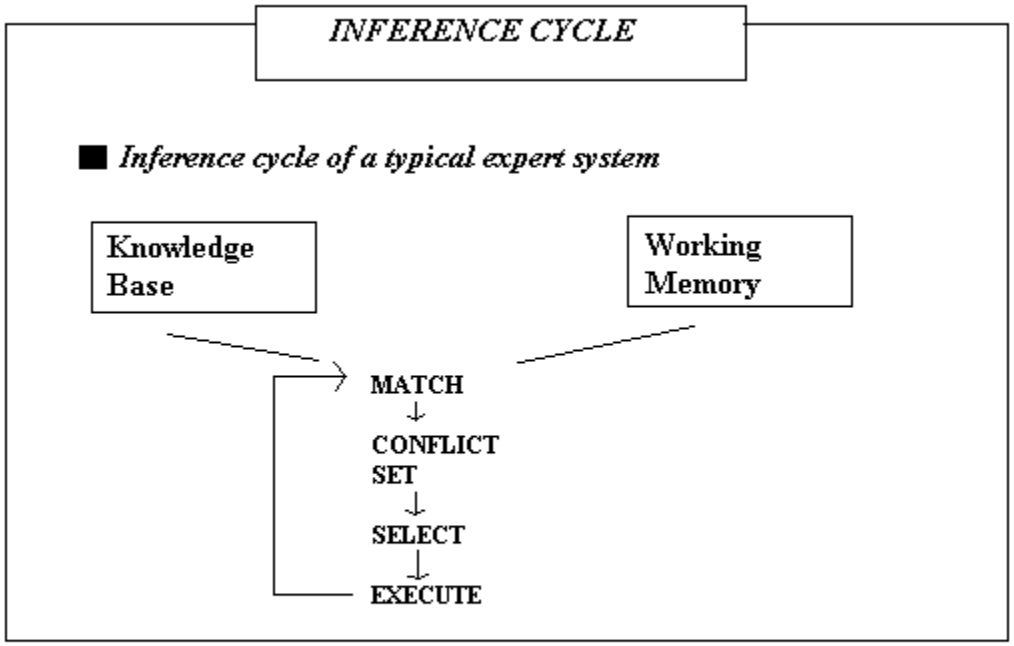
1) Mixed Chaining with priority to backward chaining
only resort to forward chaining when unable to backward chaining
Assume working memory has {A,C} ?goal to determine K.

B B B
R_1,R_3,R_8,R_4,R_7,R_5,R_6,R_2

2) Mixed Chaining with priority to forward chaining
some rule -set and goal and facts

R_8,R_4,R_7,R_5,R_6,R_2(B)

Result: 9 steps v.s +steps



CONTROL

There are two problems addressed by the inference engine:

1) It must have a way to decide where to start.Rules and facts reside in a static knowledge base. There must be a way for the reasining process to begin.

2) The inference engine must resolve conflicts that occur when alternative links of reasining emerge,The system may reach a point where there are more than a few rules ready to fire. The inference engine must choose which rule to examine next.

CONFLICT RESOLUTION STRATEGIES (A PARTIAL LIST)

- **Refractoriness-** Once a given rule fires then that same rule will be disallowed for subsequent firing.(Avoid applying a rule more than once for the same situation)
- **Recency -** Rules that apply to the most recently working elements are chosen in preference to thosewhich apply to older working elements.

- most complicated than others
- because it requires that each fact in the fact set is supplemented with a time tag , (or stamp)
- a unique number indicating the "time" the fact was derived.

Ex :

Consider the following fact set

t_1 : x=a,
t_2 : x=b,
t_3 : y=c,
t_4 : z=d

x occurs twice at time t_1 has taken value a and at time t_2 obtained the value b.

- **Specificity -** Rules which have more conditions on the left hand side are chosen in preference to those with fewer conditions.

  A rulebase can easily be enlarged by adding new, more specific rules to it without worrying too much about older ones, because more specific Production Rules will prevail over more general ones.

  For example, a person encountering a friend in the street will not be inclined to think this person is a mammal , but instead think of the person by name , just applying the most specific knowledge.

  Give priority to rules with more specific antecedents.

  A general rule : If B And D Then P And Q And L.
  A specific rule : If A and B And C And D Then J And K And L.

Knowledge Representation

What is Knowledge?

The Chambers 20th Century Dictionary provides as good a definition as any:

knowledge, assured belief; that which is known; information; …

In order to solve the complex problems encountered in AI, one generally needs a large amount of knowledge, and suitable mechanisms for representing and manipulating all that knowledge.
Knowledge can take many forms. Some simple examples are**:**

John has an umbrella
It is raining
An umbrella stops you getting wet when it's raining
An umbrella will only stop you getting wet if it is used properly
Umbrellas are not so useful when it is very windy

So, how should an AI agent store and manipulate knowledge like this?

What is a Knowledge Representation?

The object of a ***knowledge representation*** is to express knowledge in a computer tractable form, so that it can be used to enable our AI agents to perform well.

A ***knowledge representation language*** is defined by two aspects:

1. **Syntax** The syntax of a language defines which configurations of the components of the language constitute valid sentences.
2. **Semantics** The semantics defines which facts in the world the sentences refer to, and hence the statement about the world that each sentence makes.

This is a very general idea, and <u>not</u> restricted to natural language.

Suppose the language is arithmetic, then

‘ x’, ‘ $\geq$’ and ‘ y’ are *components* (or symbols or words) of the language

the *syntax* says that ‘ $x \geq y$’ is a valid sentence in the language, but ‘ $\geq \geq$ x y’ is

not the *semantics* say that ‘ $x \geq y$’ is false if y is bigger than x, and true otherwise

Requirements of a Knowledge Representation

A good knowledge representation system for any particular domain should possess the following properties:

1. **Representational Adequacy** – the ability to represent all the different kinds of knowledge that might be needed in that domain.
2. **Inferential Adequacy** –the ability to manipulate the representational structures to derive new structures (corresponding to new knowledge) from existing structures.
3. **Inferential Efficiency** – the ability to incorporate additional information into the knowledge structure which can be used to focus the attention of the inference mechanisms in the most promising directions.
4. **Acquisitional Efficiency** – the ability to acquire new information easily. Ideally the agent should be able to control its own knowledge acquisition, but direct insertion of information by a 'knowledge engineer' would be acceptable.

Finding a system that optimises these for all possible domains is not going to be feasible.

Practical Aspects of Good Representations

In practice, the theoretical requirements for good knowledge representations can usually be achieved by dealing appropriately with a number of practical requirements:

1. The representations need to be ***complete*** – so that everything that could possibly need to be represented, can easily be represented.
2. They must be ***computable*** – implementable with standard computing procedures.
3. They should make the important ***objects*** and ***relations*** explicit and accessible – so that it is easy to see what is going on, and how the various components interact.
4. They should ***suppress irrelevant detail*** – so that rarely used details don't introduce unnecessary complications, but are still available when needed.
5. They should expose any natural ***constraints*** – so that it is easy to express how one object or relation influences another.
6. They should be ***transparent*** – so you can easily understand what is being said.
7. The implementation needs to be ***concise*** and ***fast*** – so that information can be stored, retrieved and manipulated rapidly.

Components of a Good Representation

For analysis purposes it is useful to be able to break any knowledge representation down into their four fundamental components:

1. The ***lexical*** part – that determines which symbols or words are used in the representation's ***vocabulary***.

2. The ***structural*** or ***syntactic*** part – that describes the***constraints*** on how the symbols can be arranged, i.e. a grammar.

3. The ***semantic*** part – that establishes a way of associating***real world meanings*** with the representations.

4. The ***procedural*** part – that specifies the access procedures that enables ways of ***creating*** and ***modifying*** representations and ***answering questions*** using them, i.e. how we generate and compute things with the representation.

In the following we shall look at these in more detail for some specific examples.

Knowledge Representation in Natural Language

Humans usually use ***natural language*** (English, Spanish, Chinese, etc.) to represent knowledge, so why not use that to represent knowledge in our AI systems?

Advantages of Natural Language

1. It is extremely expressive – we can express virtually everything in natural language (real world situations, pictures, symbols, ideas, emotions, reasoning, …).
2. Most humans use it most of the time as their knowledge representation of choice (how many text books are not written in natural language?).

Disadvantages

1. Both the syntax and semantics are very complex and not fully understood.
2. There is little uniformity in the structure of sentences.
3. It is often ambiguous – in fact, it is*usually* ambiguous.

Database Systems

Simple ***databases*** are commonly used to good effect in Computer Science. They can be use to store and manipulate virtually any kind of information.

For example, the database may consist of a number of simple records stored in ASCII format:

```
Person record = {  name : max 32 characters
                   age : 3 digits in range 000-
                   127 sex : male or female
                   marital status : single, engaged, married, divorced,
                   widowed employer : company code of 3 characters
                   children's names : up to 8 names each with max 16 characters
                 }
```

Generally, the records can have any number of fields, containing whatever information we need, in any format, together with any appropriate links between them

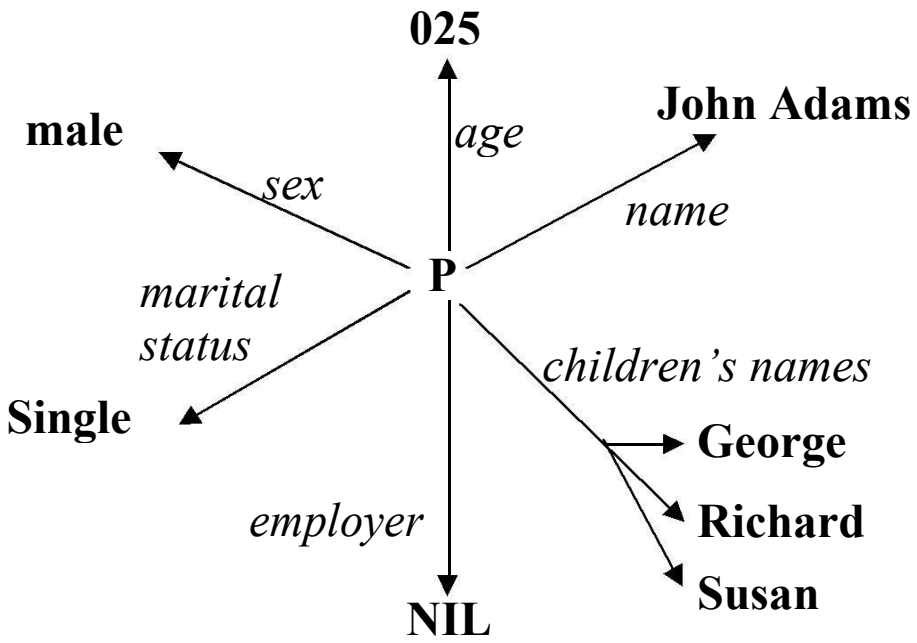
Instances in a Database System

Information in a database can be displayed in a variety of ways, for example:

A Record Structure

John Adams
025
male
single
NIL
George
Richard
Susan

A Directed Graph



But storage and display are not enough – we also need to manipulate the knowledge.

Manipulations of a Database System

We can construct sentences in an appropriate language, for example:

"marital_status(John Adams) is single"	CORRECT
"marital_status(John Adams) is divorced"	INCORRECT SEMANTICS
"marital_status(025) is male"	INCORRECT SYNTAX

We can also generate relations, for example:

R1: Employment

.	.
NIL	John Adams
NIL	Fred Smith
NIL	Sam Patel
NTL	Jo McNeal
.	.

R2: Parent/Child

.	.
John Adams	George
John Adams	Richard
John Adams	Susan
Karen Adams	Richard
.	.

Databases as a Knowledge Representation

Traditional database systems are clearly very powerful, but for AI systems they are rather limited. The important issues are:

Advantages

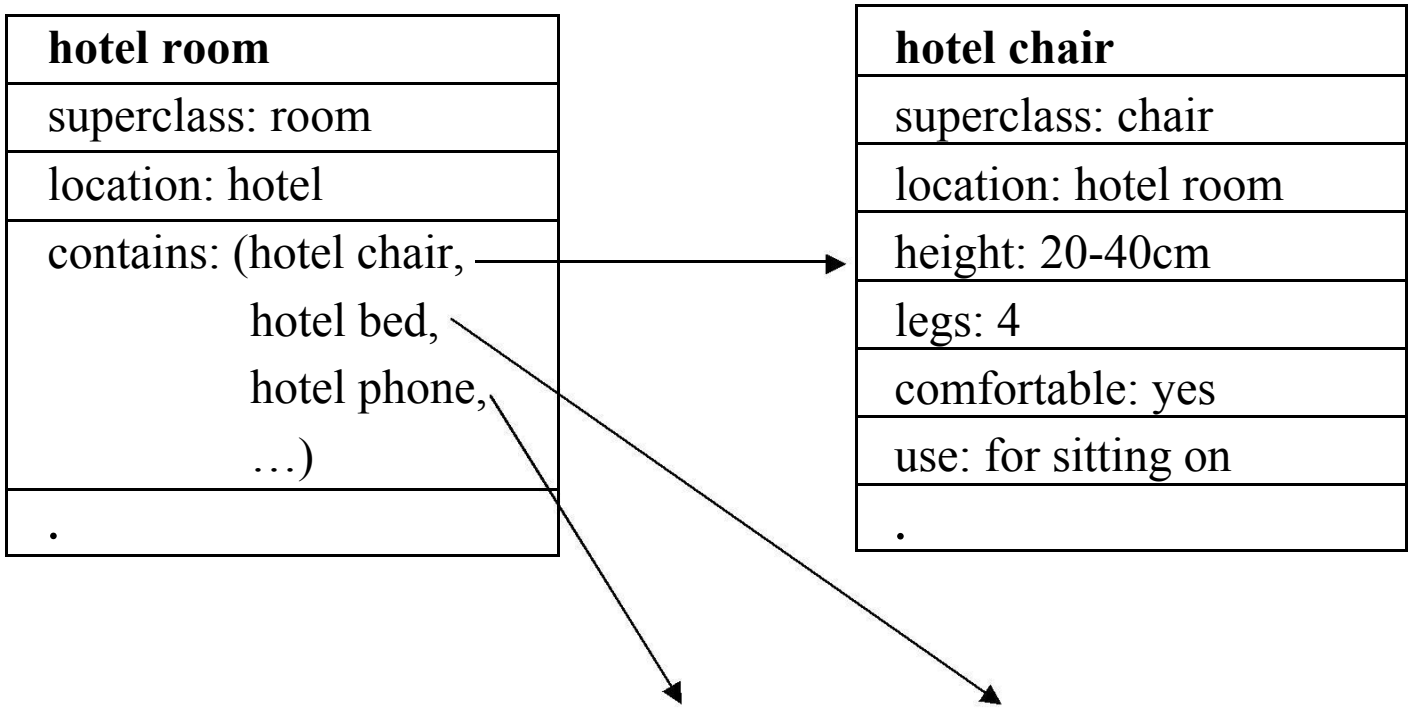
1. Databases are well suited to efficiently representing and processing large amounts of data (and derivation from a database is virtually independent of its size).
2. We can build on traditional database systems to process more complex and more powerful representational devices (e.g. frames).

Disadvantages

1. Only simple aspects of the problem domain can be accommodated.
2. We can represent *entities*, and *relationships* between entities, but not much more.
3. Reasoning is very simple – basically the only reasoning possible is simple lookup, and we usually need more sophisticated processing than that.

Frame Based Systems

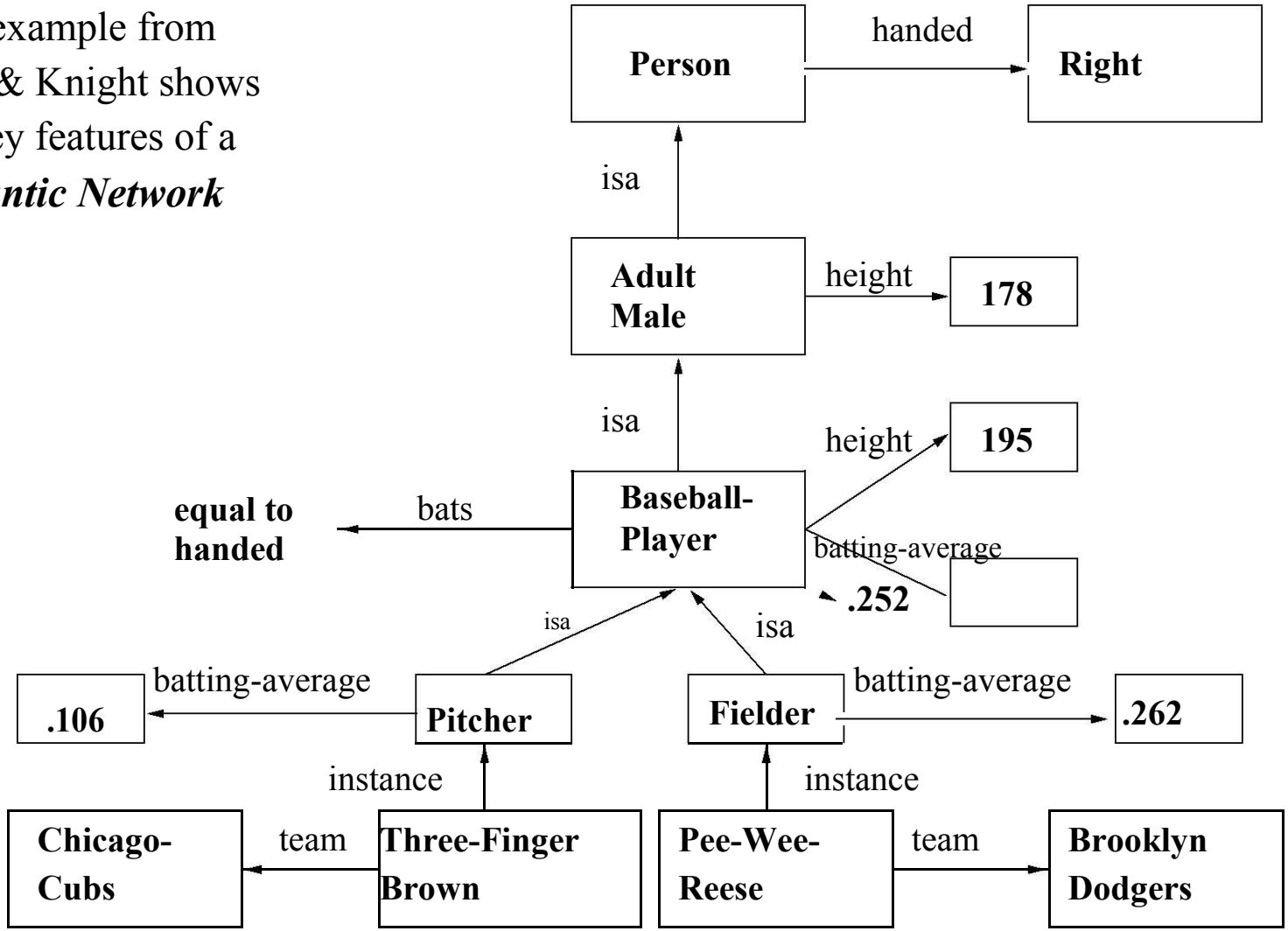
We can extend database records to ***Frames*** consisting of slots which can be filled by values, or procedures for calculating values, or pointers to other frames. For example:



Generally a whole hierarchy of frames is used to represent the required domain. It is often helpful to represent the structure of that hierarchy as a Semantic Network.

Semantic Networks

This example from Rich & Knight shows the key features of a ***Semantic Network***



First Order Logic

The syntax and semantics of **first order logic** will be covered in detail elsewhere.

Some typical sentences in first order logic are:

1. man(William) $\vee$ woman(Susan)
2. married(William, Susan)
3. $\forall x \exists y$[person(x) $\Rightarrow$ has_mother(x,y)]
4. $\forall x \forall y$[[parents(x,y) $\wedge$ man(x)] $\Rightarrow \neg$man(y)

The language consists of constants {William, Susan, etc.}, variables {x, y, etc.}, functions/predicates {Married(x,y), person(x), etc.}, and the logic symbols:

Logic	$\vee$	$\wedge$	$\Rightarrow$	$\neg$	$\forall$	$\exists$
Nat. Lang.	or	and	implies	not	for all	there exists

We can also manipulate the logic representations to generate new knowledge.

First Order Logic as a Knowledge Representation

We can combine sentences by the ‘rules of logic’ to produce new sentences, e.g.

$$\frac{\neg man(Chris) \qquad \neg man(x) \Rightarrow woman(x)}{woman(Chris)}$$

As a knowledge representation, first order logic has pros and cons:

Advantages

1. It is very expressive.
2. It has unambiguous syntax and semantics.

Disadvantage

1. There is no generally efficient procedure for processing knowledge

Rule Based Systems

A **rule based system** consists of:

1. A **database management system** for handling the domain specific facts.
2. A **rule set** for representing the knowledge structure/relations.
3. A **rule interpreter** to carry out the problem solving.

A typical rule set might be:

R1. IF Raining $\wedge$ Outside(x) $\wedge$ HasUmbrella(x) THEN UseUmbrella(x)

R2. IF Raining $\wedge$ Outside(x) $\wedge$ $\neg$HasUmbrella(x) THEN GetWet(x)

R3. IF GetsWet(x) THEN CatchCold(x)

R4. IF Sunny $\wedge$ Outside(x) THEN GetSunBurnt(x)

It should be easy enough to set up an appropriate database management system.

Rule based Inference

If we have a knowledge base consisting of **facts** and **rules**, and a rule interpreter to match the rule conditions against the facts, and a means for extracting the rules, then we can derive new knowledge. For example, using the above rule set:

Suppose we have three initial facts: Raining, Outside(John), ¬HasUmbrella(John).

Then only the rule R2 with 'x = John' matches the facts, so we are able to infer GetsWet(John). This means we now have four facts in our knowledge base: Raining, Outside(John), ¬HasUmbrella(John), GetsWet(John).

Then R3 with 'x = John' matches the facts, so we can also infer CatchesCold(John), and end up with five facts: the initial three, GetsWet(John), CatchesCold(John).

Note that there is no way we can end up with GetsSunTan(John).

The process of deriving new facts from given facts is called ***inference.***

Rule Based Systems as a Knowledge Representation

We can see that rule based systems have many of the properties required of a knowledge representation. However, as always, there are pros and cons:

Advantages

1. These systems are very expressive.
2. The rules lead to a degree of modularity.
3. We can easily introduce procedures for handling certainty factors, and this leads to the possibility of probabilistic reasoning.

Disadvantage

1. There is a lack of precise semantics for the rules.
2. The systems are not always efficient.

We shall study rule based systems in detail in a series of lectures later in this module.

Which Knowledge Representation is Best?

We have now seen what is required of a knowledge representation and taken a brief tour through a number of the most obviously plausible styles of knowledge representation.

There are clearly many more representational formalisms that might be useful. For a start, we have only really considered ***symbolic*** representations. There also exist ***non-symbolic*** (e.g. pictorial) representations. So-called ***sub-symbolic*** representations are also possible (e.g. as one finds in the activation patterns of neural network systems).

Probabilistic Reasoning and Bayesian Belief Networks:-

Probability of an Event

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

Let sample space S be the set of all possible outcomes
Let Event A be any subset of S

Definition 1: probability(A) = (number of outcomes in A)/ (total number of outcomes) P(A) = |A| / |S|

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

P(A) is called prior (unconditional) probability of A
P(~A) is the probability event A not to take place.

Example 1: the probability to pick a spade card out of a deck of 52 cards is 13/52 = ¼ The probability to pick an Ace out of a deck of 52 cards is 4/52 = 1/13

Probability Axioms:

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

(2) P(A) = 1 – P(~A)

(3) P(A v B) = P(A) + P(B) – P(A & B)

P(A v B) means the probability of either A or B or both to be true
P(A&B) means the probability of both A and B to be true.

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

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

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

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

2. Random Variables and Probability Distributions

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

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

Let X be a random variable with a domain $\langle x_1, x_2, \ldots, x_n \rangle$
The probability distribution of X is denoted by $\mathbf{P}(X) = \langle P(X = x_1), P(X=x_2), \ldots, P(X=x_n) \rangle$ Note that $P(X = x_1) + P(X = x_2) + \ldots + P(X = x_n) = 1$

Example 5: Let Weather be a random variable with values <sunny, cloudy, rainy, snowy>
Assume that records for some town show that in a year 100 days are rainy, 50 days are snowy, 120 days are cloudy (but without snow or rain) and 95 days are sunny.

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

Thus **P**(Weather) = <0.26, 0.33, 0.27, 0.14> is the probability distribution of the random variable Weather.

3. Joint Distributions

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

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

The following table illustrates the example:

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

Let H be the subset of S of individuals coming from high-income families, |H| = 600
Let C be the subset of S of individuals that have college education, |C| = 670

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

P(H) = 600 / 1000 = 0.6 (60%) P(~H) = 400 / 1000 = 0.4 (40%)

P(C) = 670 / 1000 = 0.67 (67%) P(~C) = 330 / 1000 = 0.33 (33%)

We can compute also P(H&C), P(H & ~C), P(~H & C), P(~H & ~C)

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

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

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

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

Thus we come to the following table:

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

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

P(C = yes, H = yes) = 0.57

Formally, joint distribution is defined as follows:

Definition 2: Let X1, X2, .., Xn be a set of random variables each with a range of specific values.

P(X1,X2,…,Xn) is called **joint distribution** of the variables X1, X2, …, Xn and it is defined by a n-dimensional table, where each cell corresponds to one particular assignment of values to the variables X1, X2, …, Xn

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

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

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

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

Given a joint distribution table we can compute prior probabilities:

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

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

4. Conditional Probabilities

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

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

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

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

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

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

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

Therefore

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

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

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

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

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

In our example, P(C&H) = P(C = yes, H = yes). Thus given the joint distribution **P** (H, C), we can compute the prior probability P(H), P(~H), P(C), P(~C) and then the conditional probability P(C|H), P(C|~H), P(H|C), P(H|~C) .

Independent events

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

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

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

The proof follows directly from Definition 3 and Definition 4.

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

Bayes' Theorem:-

From Definition 3 we have

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

However, P(A&B) = P(B&A)
Therefore

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

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

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

More than 2 variables

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

$$P(X|Y1\&Y2 \& \ldots \& Yn) = P(Y1 \& Y2 \& \ldots \& Yn \mid X) * P(X) / P(Y1 \& Y2 \& \ldots \& Yn)$$

Think of X as being a hypothesis, and Y1, Y2, …, Yn as being n pieces of evidence for the hypothesis. When Y1, Y2, …, Yn are independent on each other, the formula takes the form:

$$P(X|Y1\&Y2 \& \ldots \& Yn) = \frac{P(Y1|X)*P(Y2|X)*...*P(Yn \mid X) * P(X)}{P(Y1)*P(Y2)*...*P(Yn)}$$

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

$$P(X1 \& X2 \& \ldots \& Xn) = P(X1) * P(X2|X1) * P(X3 \mid X2 \& X1) \ldots P(Xn \mid Xn\text{-}1 \& \ldots X1)$$

Normalization

Consider the probability of malaria given headache

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

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

We have:

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

Adding these equations we obtain

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

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

So we have

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

Multiply both sides by P(H):

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

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

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

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

Relative Likelihood of two events

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

P(plague|headache) = P(headache | plague) * P(plague) / P(headache)
P(flu | headache) = P(headache | flu) * P(flu) / P(headache)

The ratio

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

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

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

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

Example: The Monty Hall game

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

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

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

Let O be the event that Monty Hall opens door B.
The Monty Hall Problem can be restated as follows: is $P(PA \mid O) = P(PC \mid O)$

By the Bayes' Theorem we have:

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

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

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

$P(O \mid PA) = 1/2$, if the car is behind A, Monty Hall can open either B or C
$P(O \mid PB) = 0$, if the car is behind B, Monty Hall will not open B
$P(O \mid PC) = 1$, if the car is behind C, Monty Hall can only open door B

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

$P(O) = 1/3 * (1/2 + 0 + 1) = 1/2$
Therefore we obtain:

$P(PA \mid O) = (1 / 2 * 1 / 3) / (1 / 2) = 1/3$
$P(PC \mid O) = (1 * 1/3) / (1 / 2) = 2/3$

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

Useful expressions

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

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

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

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

Simple Bayesian Concept Learning

The Bayes' theorem can be used to solve the following problem:
Determine the most probable hypothesis out of *n* possible hypotheses H_1, H_2 , .., H_n , given a set of evidence E. For each H_i we can compute

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

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

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

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

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

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

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

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

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

First-order logic

- First-order logic (FOL) models the world in terms of
  - **Objects,**:which are things with individual identities
  - **Properties** :of objects that distinguish them from other objects
  - **Relations** :that hold among sets of objects
  - **Functions:**which are a subset of relations where there is only one "value" for any given "input"
- Examples:
  - Objects: Students, lectures, companies, cars ...
  - Relations: Brother-of, bigger-than, outside, part-of, has-color, occurs-after, owns, visits, precedes, ...
  - Properties: blue, oval, even, large, ...
  - Functions: father-of, best-friend, second-half, one-more-than ...

User provides

- **Constant symbols,** which represent individuals in the world
  - Mary
  - 3
  - Green
- **Function symbols,** which map individuals to individuals
  - father-of(Mary) = John
  - color-of(Sky) = Blue
- **Predicate symbols,** which map individuals to truth values
  - greater(5,3)
  - green(Grass)
  - color(Grass, Green)

FOL Provides

- **Variable symbols**
  - E.g., x, y, foo
- **Connectives**
  - Same as in PL: not ($\neg$), and ($\wedge$), or ($\vee$), implies ($\rightarrow$), if and only if (biconditional $\leftrightarrow$)
- **Quantifiers**
  - Universal $\forall$**x** or **(Ax)**
  - Existential $\exists$**x** or **(Ex)**

Sentences are built from terms and atoms

- A **term** (denoting a real-world individual) is a constant symbol, a variable symbol, or an n-place function of n terms.

  x and $f(x_1, ..., x_n)$ are terms, where each x_i is a term. A term with no variables is a **ground term**

- An **atomic sentence** (which has value true or false) is an n-place predicate of n terms
- A **complex sentence** is formed from atomic sentences connected by the logical connectives:

  $\neg P$, $P \vee Q$, $P \wedge Q$, $P \rightarrow Q$, $P \leftrightarrow Q$ where P and Q are sentences

- A **quantified sentence** adds quantifiers $\forall$ and $\exists$
- A **well-formed formula (wff)** is a sentence containing no "free" variables. That is, all variables are "bound" by universal or existential quantifiers.

  $(\forall x)P(x,y)$ has x bound as a universally quantified variable, but y is free.

Quantifiers

- **Universal quantification**
  - $(\forall x)P(x)$ means that P holds for **all** values of x in the domain associated with that variable
  - E.g., $(\forall x)$ dolphin(x) $\rightarrow$ mammal(x)
- **Existential quantification**
  - $(\exists\ x)P(x)$ means that P holds for **some** value of x in the domain associated with that variable
  - E.g., $(\exists\ x)$ mammal(x) $\wedge$ lays-eggs(x)
  - Permits one to make a statement about some object without naming it

Quantifier Scope

- Switching the order of universal quantifiers *does not* change the meaning:
  - $(\forall x)(\forall y)P(x,y) \leftrightarrow (\forall y)(\forall x)\ P(x,y)$
- Similarly, you can switch the order of existential quantifiers:
  - $(\exists x)(\exists y)P(x,y) \leftrightarrow (\exists y)(\exists x)\ P(x,y)$
- Switching the order of universals and existentials *does* change meaning:
  - $(\forall x)(\exists y)$ likes(x,y)
  - $(\exists y)(\forall x)$ likes(x,y)

Connections between All and Exists

We can relate sentences involving $\forall$ and $\exists$ using De Morgan's laws:

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

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

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

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

Translating English to FOL

Every gardener likes the sun.

You can fool some of the people all of the time. You can fool all of the people some of the time. All purple mushrooms are poisonous.

No purple mushroom is poisonous.

There are exactly two purple mushrooms.

Clinton is not tall.

X is above Y iff X is on directly on top of Y or there is a pile of one or more other objects directly on top of one another starting with X and ending with Y.

Translating English to FOL

Every gardener likes the sun.

$\forall x$ gardener(x) $\rightarrow$ likes(x,Sun)

You can fool some of the people all of the time.

$\exists x \; \forall t$ person(x) $\wedge$time(t) $\rightarrow$ can-fool(x,t)

You can fool all of the people some of the time.

$\forall x \; \exists t$ (person(x) $\rightarrow$ time(t) $\wedge$can-fool(x,t)) $\leftarrow$ Equivalent $\forall x$ (person(x) $\rightarrow \exists t$ (time(t) $\wedge$can-fool(x,t))

All purple mushrooms are poisonous.

$\forall x$ (mushroom(x) $\wedge$ purple(x)) $\rightarrow$ poisonous(x)

No purple mushroom is poisonous.

$\neg\exists x$ purple(x) $\wedge$ mushroom(x) $\wedge$ poisonous(x)

$\forall x$ (mushroom(x) $\wedge$ purple(x)) $\rightarrow \neg$poisonous(x)

$\leftrightarrows$ Equivalent

There are exactly two purple mushrooms.

$\exists x \; \exists y$ mushroom(x) $\wedge$ purple(x) $\wedge$ mushroom(y) $\wedge$ purple(y) ^ $\neg$(x=y) $\wedge \forall z$ (mushroom(z) $\wedge$ purple(z)) $\rightarrow$ ((x=z) $\vee$ (y=z))

Clinton is not tall.

$\neg$tall(Clinton)

X is above Y iff X is on directly on top of Y or there is a pile of one or more other objects directly on top of one another starting with X and ending with Y.

$\forall x \; \forall y$ above(x,y) $\leftrightarrow$ (on(x,y) $\vee \exists z$ (on(x,z) $\wedge$ above(z,y)))

Example: A simple genealogy KB by FOL

- **Build a small genealogy knowledge base using FOL that**
  - contains facts of immediate family relations (spouses, parents, etc.)
  - contains definitions of more complex relations (ancestors, relatives)
  - is able to answer queries about relationships between people
- **Predicates:**
  - parent(x, y), child(x, y), father(x, y), daughter(x, y), etc.
  - spouse(x, y), husband(x, y), wife(x,y)
  - ancestor(x, y), descendant(x, y)
  - male(x), female(y)
  - relative(x, y)
- **Facts:**
  - husband(Joe, Mary), son(Fred, Joe)
  - spouse(John, Nancy), male(John), son(Mark, Nancy)
  - father(Jack, Nancy), daughter(Linda, Jack)
  - daughter(Liz, Linda)
  - etc.

- **Rules for genealogical relations**
  - $(\forall x,y)$ parent(x, y) $\leftrightarrow$ child (y, x)

    $(\forall x,y)$ father(x, y) $\leftrightarrow$ parent(x, y) $\wedge$ male(x) (similarly for mother(x, y))
    $(\forall x,y)$ daughter(x, y) $\leftrightarrow$ child(x, y) $\wedge$ female(x) (similarly for son(x, y))
  - $(\forall x,y)$ husband(x, y) $\leftrightarrow$ spouse(x, y) $\wedge$ male(x) (similarly for wife(x, y))
    $(\forall x,y)$ spouse(x, y) $\leftrightarrow$ spouse(y, x) (**spouse relation is symmetric**)
  - $(\forall x,y)$ parent(x, y) $\rightarrow$ ancestor(x, y)
    $(\forall x,y)(\exists z)$ parent(x, z) $\wedge$ ancestor(z, y) $\rightarrow$ ancestor(x, y)
  - $(\forall x,y)$ descendant(x, y) $\leftrightarrow$ ancestor(y, x)
  - $(\forall x,y)(\exists z)$ ancestor(z, x) $\wedge$ ancestor(z, y) $\rightarrow$ relative(x, y)
    (related by common ancestry)

    $(\forall x,y)$ spouse(x, y) $\rightarrow$ relative(x, y) (related by marriage)
    $(\forall x,y)(\exists z)$ relative(z, x) $\wedge$ relative(z, y) $\rightarrow$ relative(x, y)
    (**transitive**) $(\forall x,y)$ relative(x, y) $\leftrightarrow$ relative(y, x) (**symmetric**)
- **Queries**
  - ancestor(Jack, Fred) /* the answer is yes */
  - relative(Liz, Joe) /* the answer is yes */
  - relative(Nancy, Matthew)
    /* no answer in general, no if under closed world assumption */
  - $(\exists z)$ ancestor(z, Fred) $\wedge$ ancestor(z, Liz)

Semantics of FOL

- **Domain M:** the set of all objects in the world (of interest)
- **Interpretation I:** includes
  - Assign each constant to an object in M
  - Define each function of n arguments as a mapping M^n => M
  - Define each predicate of n arguments as a mapping M^n => {T, F}
  - Therefore, every ground predicate with any instantiation will have a truth value
  - In general there is an infinite number of interpretations because |M| is infinite
- **Define logical connectives: ~, ^, v, =>, <=>** as in PL
- **Define semantics of (∀x) and (∃x)**
  - (∀x) P(x) is true iff P(x) is true under all interpretations
  - (∃x) P(x) is true iff P(x) is true under some interpretation

- **Model:** an interpretation of a set of sentences such that every sentence is *True*
- **A sentence is**
  - **satisfiable** if it is true under some interpretation
  - **valid** if it is true under all possible interpretations
  - **inconsistent** if there does not exist any interpretation under which the sentence is true
- **Logical consequence:** S |= X if all models of S are also models of X

Resolution

- Reminder: Resolution rule for propositional logic:
  - $P_1 \vee P_2 \vee ... \vee P_n$
  - $\neg P_1 \vee Q_2 \vee ... \vee Q_m$
  - Resolvent: $P_2 \vee ... \vee P_n \vee Q_2 \vee ... \vee Q_m$
- Examples
  - P and $\neg$ P $\vee$ Q : derive Q (Modus Ponens)
  - ($\neg$ P $\vee$ Q) and ($\neg$ Q $\vee$ R) : derive $\neg$ P $\vee$ R
  - P and $\neg$ P : derive False [contradiction!]
  - (P $\vee$ Q) and ($\neg$ P $\vee \neg$ Q) : derive True

Resolution in first-order logic

- Resolution is **sound** and **complete** for FOL
- Given sentences

  $P_1 \vee ... \vee P_n$ and $Q_1 \vee ... \vee Q_m$
- in *conjunctive normal form:*
  - each P_i and Q_i is a literal, i.e., a positive or negated predicate symbol with its terms,
- if P_j and $\neg Q_k$ **unify** with substitution list θ, then derive the resolvent sentence:

  $subst(\theta, P_1 \vee ... \vee P_{j-1} \vee P_{j+1} ... P_n \vee Q_1 \vee \ldots Q_{k-1} \vee Q_{k+1} \vee ... \vee Q_m)$
- Example
  - from clause $\mathbf{P(x, f(a)) \vee P(x, f(y)) \vee Q(y)}$
  - and clause $\mathbf{\neg P(z, f(a)) \vee \neg Q(z)}$
  - derive resolvent $\mathbf{P(z, f(y)) \vee Q(y) \vee \neg Q(z)}$
  - using $\mathbf{\theta = \{x/z\}}$

Resolution refutation

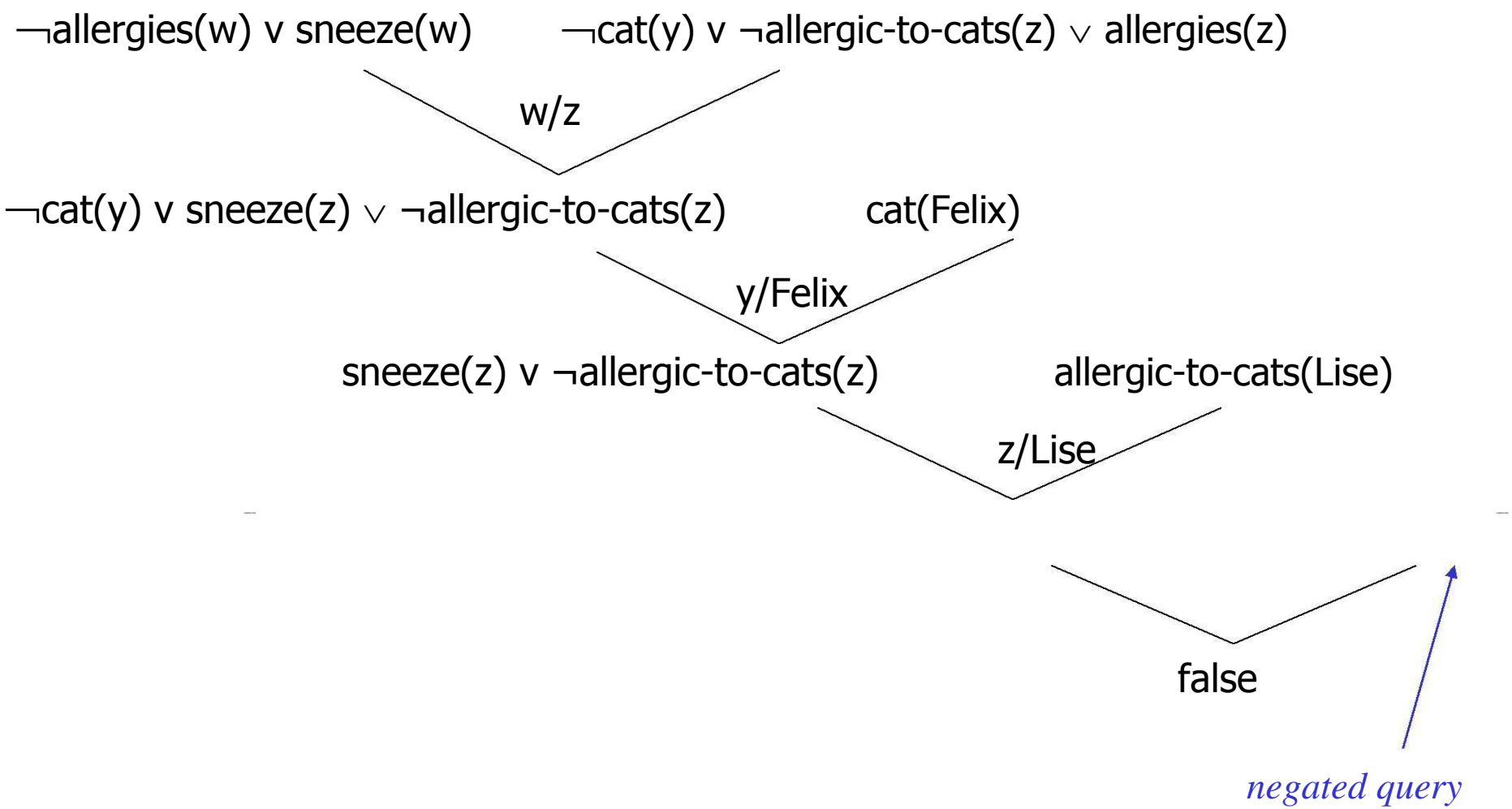
- Given a consistent set of axioms KB and goal sentence Q, show that KB $\models$ Q
- **Proof by contradiction:** Add $\neg$Q to KB and try to prove false.

  i.e., (KB $\vdash$ Q) $\leftrightarrow$ (KB $\wedge$ $\neg$Q $\vdash$ False)
- Resolution is **refutation complete:** it can establish that a given sentence Q is entailed by KB, but can't (in general) be used to generate all logical consequences of a set of sentences

Resolution example

- KB:
  - allergies(X) $\rightarrow$ sneeze(X)
  - cat(Y) $\wedge$ allergic-to-cats(X) $\rightarrow$ allergies(X)
  - cat(Felix)
  - allergic-to-cats(Lise)
- Goal:
  - sneeze(Lise)

Refutation resolution proof tree



UTILITY THEORY

A CLASSIFICATION OF DECISION MAKING

Decision under Certainty

> Definition 1. We say that the decision is taken under certainty if each action is known to lead invariably to a specific outcome (prospect, alternative, etc.).

Mathematical tools: the calculus to find maxima and minima of functions, the calculus of variations to find functions, production schedules, inventory schedules, etc.

Decision under Risk

> Definition 2. We say that the decision is taken under risk if each action leads to one of a set of possible specific outcomes, each outcome occurring with a known probability.

Remark. Certainty is a degenerate case of risk where the probabilities are 0 and 1.

Example 1. An action might lead to a reward of \$10 if a fair coin comes up heads, and a loss of \$5 if it comes up tails.

Example 2. More generally, consider a gamble in which one of n outcomes will occur, and let the possible outcomes be worth $a_1, a_2, \ldots a_n$ euros, respectively. Suppose that it is known that the respective probabilities of these outcomes are $p_1, p_2, \ldots, p_n$, where each p_i lies between 0 and 1 (inclusive) and their sum is 1. How much is it worth to participate in this gamble?

The monetary expected value: $b = a_1p_1 + a_2p_2 + \cdots + a_np_n$.

Objections to the monetary expected value – St. Petersburg Paradox:

Peter tosses a coin and continues to do so until it should land ”heads” when it comes to the ground. He agrees to give Paul one ducat if he gets ”heads” on the very first throw, two ducats if he gets it on the second, four if on the third, eight if on the fourth, and so on, so that with each additional throw the number of ducats he must pay is doubled. Suppose we seek to determine the value of Paul’s expectation.

The mean value of the win in ducats:

$$1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{2^2} + 2^2 \cdot \frac{1}{2^3} + \cdots + 2^n \cdot \frac{1}{2^{n+1}} + \cdots = \infty$$

Paradox: a reasonable person sells – with a great pleasure – the engagement in the play for 20 ducats.

Daniel Bernoulli: a gamble should be evaluated not in terms of the value of its alternative pay-offs but rather in terms of the value of its utilities, which he derived to be logarithmic functions.

Decision under Uncertainty

> Definition 3. We say that the decision is taken under uncertainty if either action has as its consequence a set of possible specific outcomes, but the probabilities of these outcomes are completely unknown or are not even meaningful.

AXIOMATIC UTILITY THEORY

Rational Preferences

Consider a finite set $\{A_1, A_2, \ldots, A_r\}$ of basic alternatives or prizes. A lottery

$$(p_1A_1, p_2A_2, \ldots, p_r A_r)$$

is a chance mechanism which yields the prizes $A_1, A_2, \ldots, A_r$ as outcomes with known probabilities $p_1, p_2, \ldots, p_r$, where each $p_i \geq 0$, $p_1 + p_2 \cdots + p_r = 1$. Let us order the alternatives downwards from the most to the least preferred one.

Among the basic alternatives, we use the symbolism $A_i \precsim A_j$ to denote that A_j is not preferred to A_i. Equivalently, we say that A_i is preferred or indifferent to A_j.

Assumption 1 (ordering of alternatives). The "preference or indifference" ordering over all basic alternatives is complete and transitive: for any A_i and A_j, either $A_i \precsim A_j$ or $A_j \precsim A_i$ holds; and if $A_i \precsim A_j$ and $A_j \precsim A_k$ then $A_i \precsim A_k$.

Now suppose that $L^{(1)}, L^{(2)}, \ldots, L^{(s)}$ are any s lotteries which each involve $A_1, A_2, \ldots, A_r$ as prizes. If $q_1, q_2, \ldots, q_r$ are any s nonnegative numbers which sum to 1, then

$$[q_1L^{(1)}, q_2L^{(2)}, \ldots, q_sL^{(s)}]$$

denotes a compound lottery in the following sense: one and only one of the given s lotteries will be the prize, and the probability that it will be $L^{(i)}$ is q_i.

For the sake of simplification, let us denote A_1 the most preferred alternative, A_r the least preferred one.

Assumption 2 (reduction of compound lotteries). Any compound lottery is indif-ferent to a simple lottery with $A_1, A_2, \ldots, A_r$ as prizes, their probabilities being computed according to the ordinary probability calculus. In particular, if

$$L^{(i)} = (p_1^{(i)}A_1, p_2^{(i)}A_2, \ldots, p_r^{(i)}A_r), \qquad \text{for } i = 1, 2, \ldots, s,$$

then

$$^{i}q_1L^{(1)}, q_2L^{(2)}, \ldots, q_sL^{(s)\not\in} \sim (p_1A_1, p_2A_2, \ldots, p_r A_r),$$

where

$$p_i = q_1p^{(1)}{}_i + q_2p^{(2)}{}_i + \cdots + q_sp^{(s)}_i.$$

Assumption 3 (continuity). Each prize A_i is indifferent to some lottery involving just A_1 and A_r. That is, there exists a number u_i such that A_i is indifferent to

$$(u_iA_1, 0A_2, \ldots, 0A_{r-1}, (1 - u_i)A_r).$$

For convenience, we write:

$$A_i \sim (u_iA_1, (1 - u_i)A_r) = \tilde{A}_i.$$

Assumption 4 (substitutibility). In any lottery L, A_i is substitutable for A_i, that is,

$$(\ ) \sim (\ \tilde{}\ )\ p_1A_1, p_2A_2, \ldots, p_iA_i, \ldots, p_r A_r\ p_1A_1, p_2A_2, \ldots, p_iA_i, \ldots, p_r A_r.$$

Assumption 5 (transitivity). Preference and indifference among lotteries are transi-tive relations.

Assumption 6 (monotonicity). A lottery $(pA_1, (1 - p)A_r)$ is preferred or indifferent to $(p^0A_1, (1 - p^0)A_r)$ if and only if $p \geq p^0$.

Theorem 1. If the preference or indifference relation % satisfies assumptions 1 trough 6, there are numbers u_i associated with the basic prizes A_i such that for two lotteries L and L^0 the magnitudes of the expected values

$$p_1u_1 + p_2u_2 + \cdots + p_r u_r \quad \text{and} \quad p^0{}_1u_1 + p^0{}_2u_2 + \cdots + p^0{}_r u_r$$

reflect the preference between the lotteries.

Definition 4. If a person imposes a transitive preference relation % over a set of lotteries and if to each lottery L there is assigned a number u(L) such that the magnitudes of the numbers reflect the preferences, i.e., $u(L) \geq u(L^0)$ if and only if $L \% L^0$, then we say there exists a utility function u over the lotteries.

If, in addition, the utility function has the property that

$$u\ (qL, (1 - q)L^0) = qu(L) + (1 - q)u(L^0)$$

for all probabilities q and lotteries L and L^0, then we say the utility function is linear.