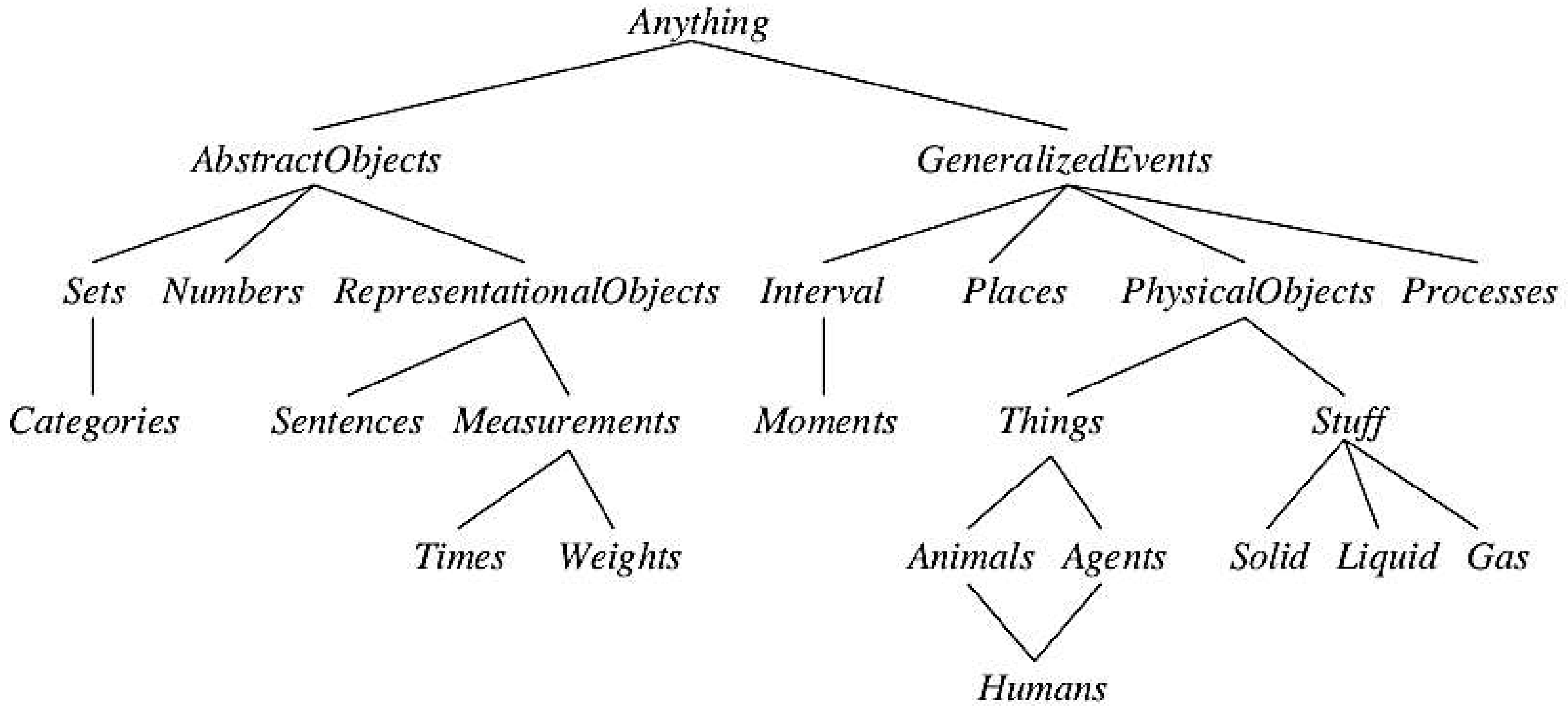
Ontological Engineering

- ✓ Knowledge is the information about a domain that is used to solve problems of a domain.
- ✓ As part of designing a program to solve problems, we must define how the knowledge is represented.
- ✓ Complex domains require more general and flexible representations of concepts like events, time, physical object and beliefs of a domain.
- ✓ Representing these abstract concepts is called **Ontological Engineering**
- ✓ The general frame work of concepts is called **upper ontology,** because of the convention of drawing graphs with the general concepts at the top and the more specific concepts below them, as in Figure

Upper Ontology



General representation of ontology of world

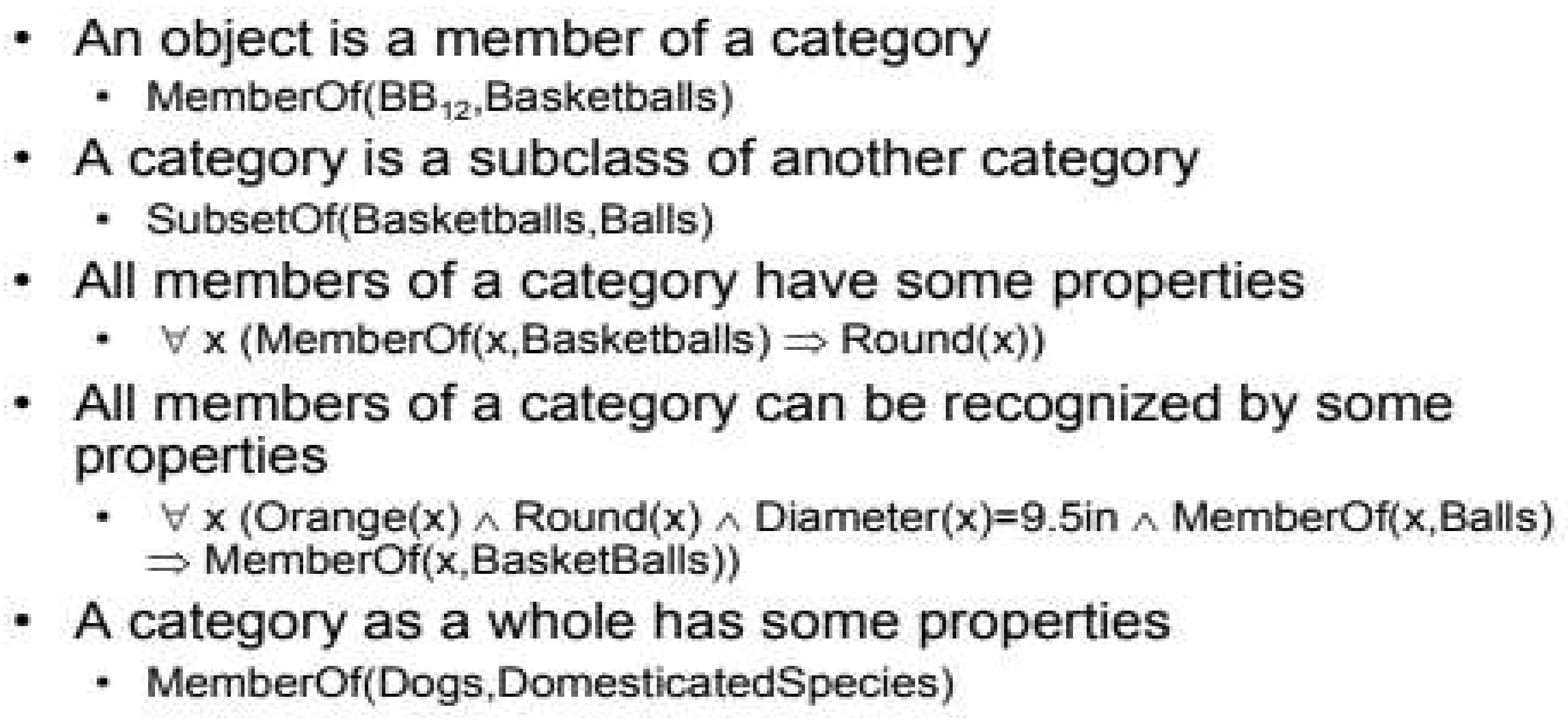
CATEGORIES AND OBJECTS

- The organization of objects into **categories** is a vital part of knowledge representation.
- Although interaction with the world takes place at the level of individual objects, *much reasoning takes place at the level of categories*.
- For example, a shopper would normally have the goal of buying a basketball, rather than a *particular* basketball such as BB9.
- Categories also serve to make predictions about objects once they are classified.
- There are two ways of representing categories in first-order logic:
  - Predicates : eg.Basketball (b),
  - Objects: eg.Basketballs

- Categories serve to organize and simplify the knowledge base through **inheritance**.
- If we say that all instances of the category Food are edible, and if we assert that Fruit is a subclass of Food and Apples is a subclass of Fruit, then we can infer that every apple is edible.
- We say that the individual apples **inherit** the property of edibility, in this case from their membership in the Food category.

First-order logic and categories

- First-order logic makes it easy to state facts about categories, either by relating objects to categories or by quantifying over their members.



- An object is a member of a category
  - MemberOf(BB_{12},Basketballs)
- A category is a subclass of another category
  - SubsetOf(Basketballs,Balls)
- All members of a category have some properties
  - $\forall$ x (MemberOf(x,Basketballs) $\Rightarrow$ Round(x))
- All members of a category can be recognized by some properties
  - $\forall$ x (Orange(x) $\wedge$ Round(x) $\wedge$ Diameter(x)=9.5in $\wedge$ MemberOf(x,Balls) $\Rightarrow$ MemberOf(x,BasketBalls))
- A category as a whole has some properties
  - MemberOf(Dogs,DomesticatedSpecies)

Relations between categories

- Two or more categories are ***disjoint*** if they have no members in common:
  - Disjoint(s)$\Leftrightarrow(\forall\ c_1,c_2\ \ c_1 \in s \wedge c_2 \in s \wedge c_1 \neq c_2 \Rightarrow$ Intersection$(c_1,c_2) = \varnothing)$
  - Example; Disjoint({animals, vegetables})
- A set of categories *s* constitutes an ***exhaustive decomposition*** of a category *c* if all members of the set *c* are covered by categories in *s*:
  - E.D.(s,c) $\Leftrightarrow (\forall\ i\ \ i \in c \Rightarrow \exists\ c_2\ \ c_2 \in s \wedge i \in c_2)$
  - Example: ExhaustiveDecomposition ({Americans, Canadian, Mexicans},NorthAmericans)

Relations between categories

- A ***partition*** is a disjoint exhaustive decomposition:
  - Partition(s,c) $\Leftrightarrow$ Disjoint(s) $\wedge$ E.D.(s,c)
  - Example: Partition({Males,Females},Persons).
- Is ({Americans,Canadian, Mexicans}, NorthAmericans) a partition? • No! There might be dual citizenships.
- Categories can be defined by providing necessary and sufficient conditions for membership
  - $\forall$ x Bachelor(x) $\Leftrightarrow$ Male(x) $\wedge$ Adult(x) $\wedge$ Unmarried(x)

Natural kinds

- Many categories have no clear-cut definitions, e.g. chair, bush, book. $\rightarrow$ ***natural kinds***
- Tomatoes: sometimes green, red, yellow, black. Mostly round.
  - We can write down useful facts about categories without providing exact definitions. $\rightarrow$ ***Prototypes***
  - category *Typical(Tomatoes)*
  - $\forall$ x, x $\in$ Typical(Tomatoes) $\Rightarrow$ Red(x) $\wedge$ Spherical(x).

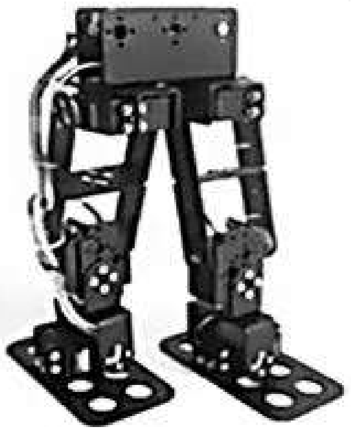
Physical composition

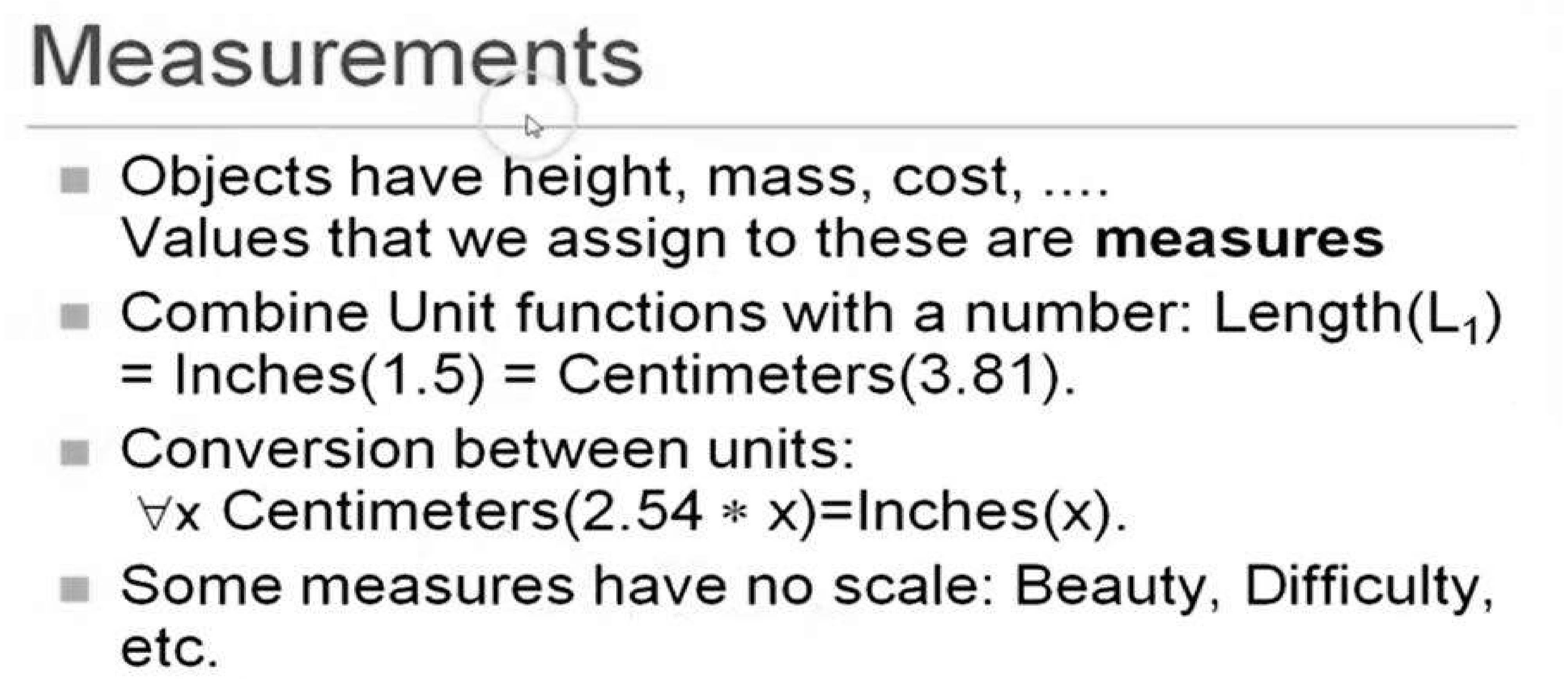
- One object may be part of another:
  - PartOf(Bucharest,Romania)
  - PartOf(Romania,EasternEurope)
  - PartOf(EasternEurope,Europe)
- The PartOf predicate is transitive (and irreflexive), so we can infer that PartOf(Bucharest,Europe)
- More generally:
  - $\forall x \;\; \mathrm{PartOf}(x,x)$
  - $\forall x,y,z \; \mathrm{PartOf}(x,y) \wedge \mathrm{PartOf}(y,z) \Rightarrow \mathrm{PartOf}(x,z)$

Physical composition

- Often characterized by structural relations among parts.
  - E.g. Biped(a) $\Rightarrow$

$$(\exists l_1, l_2, b)(Leg(l_1) \wedge Leg(l_2) \wedge Body(b) \wedge$$
$$PartOf(l_1, a) \wedge PartOf(l_2, a) \wedge PartOf(b, a) \wedge$$
$$Attached(l_1, b) \wedge Attached(l_2, b) \wedge$$
$$l_1 \neq l_2 \wedge (\forall l_3)(Leg(l_3) \Rightarrow (l_3 = l_1 \vee l_3 = l_2)))$$





Measurements

- Objects have height, mass, cost, ...
  Values that we assign to these are **measures**
- Combine Unit functions with a number: Length(L_1) = Inches(1.5) = Centimeters(3.81).
- Conversion between units:
  $\forall x$ Centimeters($2.54 * x$)=Inches(x).
- Some measures have no scale: Beauty, Difficulty, etc.

- Measures can be used to describe objects as follows:
  Diameter (Basketball)=Inches(9.5) .
  ListPrice(Basketball)=$(19) .
  $d \in$ Days $\Rightarrow$ Duration(d)=Hours(24) .

Measurements

$$e_1 \in Exercises \land e_2 \in Exercises \land Wrote(Norvig, e_1) \land Wrote(Russell, e_2) \Rightarrow$$
$$Difficulty(e_1) > Difficulty(e_2)\ .$$
$$e_1 \in Exercises \land e_2 \in Exercises \land Difficulty(e_1) > Difficulty(e_2) \Rightarrow$$
$$ExpectedScore(e_1) < ExpectedScore(e_2)\ .$$

Objects

- the real world can be seen as consisting of primitive objects(e.g., atomic particles)and composite objects built from them.
- Stuff   ex: butter
- Things ex: dog
- some properties are intrinsic: they belong to the very substance of the object, rather than to the objects as a whole
- extrinsic properties-weight, length, shape and so on -are not retained under subdivision.
- A category of objects that includes in its definition only intrinsic properties is then a substance, or mass noun; a class that includes any extrinsic properties in its definition is a count noun

- The category **Stuff** is the most general substance category, specifying no intrinsic properties.
- The category **Thing** is the most general discrete object category, specifying no extrinsic properties.
- Ex:
  - any part of a butter-object is also a butter-object:

    $b \in Butter \land PartOf(p, b) \Rightarrow p \in Butter$.
  - We can now say that butter melts at around 30 degrees centigrade:

    $b \in Butter \Rightarrow MeltingPoint(b, Centigrade(30))$

Events

- situation calculus represents actions and their effects
- situation calculus- it was designed to describe a world in which actions are discrete, instantaneous and happen one at a time.
- Event calculus-based on points of time rather than on situations
- event calculus reifies **fluents** and **events**
- the fluent
  - At(shankar, berkeley): is an object that refers to the fact of shankar being in Berkeley, but does not by itself say anything about whether it is true.
- To assert that a fluent is actually true at some point in time we use the predicate T, as in
- T(At(shankar, Berkeley),t).

- Events are described as instances of event categories
- The event E1 of shankar flying from San Francisco to Washington D.C. is described as
- E1 ∈ Flyings ∧ Flyer (E1, Shankar) ∧ Origin(E1, SF) ∧ Destination(E1,DC) .
- we can define an alternative three-argument version of the category of flying events and say
- E1 ∈ Flyings(Shankar , SF,DC) .

- The complete set of predicates for one version of the event calculus is
- T(f, t) Fluent f is true at time t
- Happens(e, i) Event e happens over the time interval i
- Initiates(e, f, t) Event e causes fluent f to start to hold at time t
- Terminates(e, f, t) Event e causes fluent f to cease to hold at time t
- Clipped(f, i) Fluent f ceases to be true at some point during time interval i
- Restored (f, i) Fluent f becomes true sometime during time interval I
- For example, we can say that the only way a wumpus-world agent gets an arrow is at the start, and the only way to use up an arrow is to shoot it:
  - Initiates(e, HaveArrow(a), t) ⇔ e = Start
  - Terminates(e, HaveArrow(a), t) ⇔ e ∈ Shootings(a)

Process

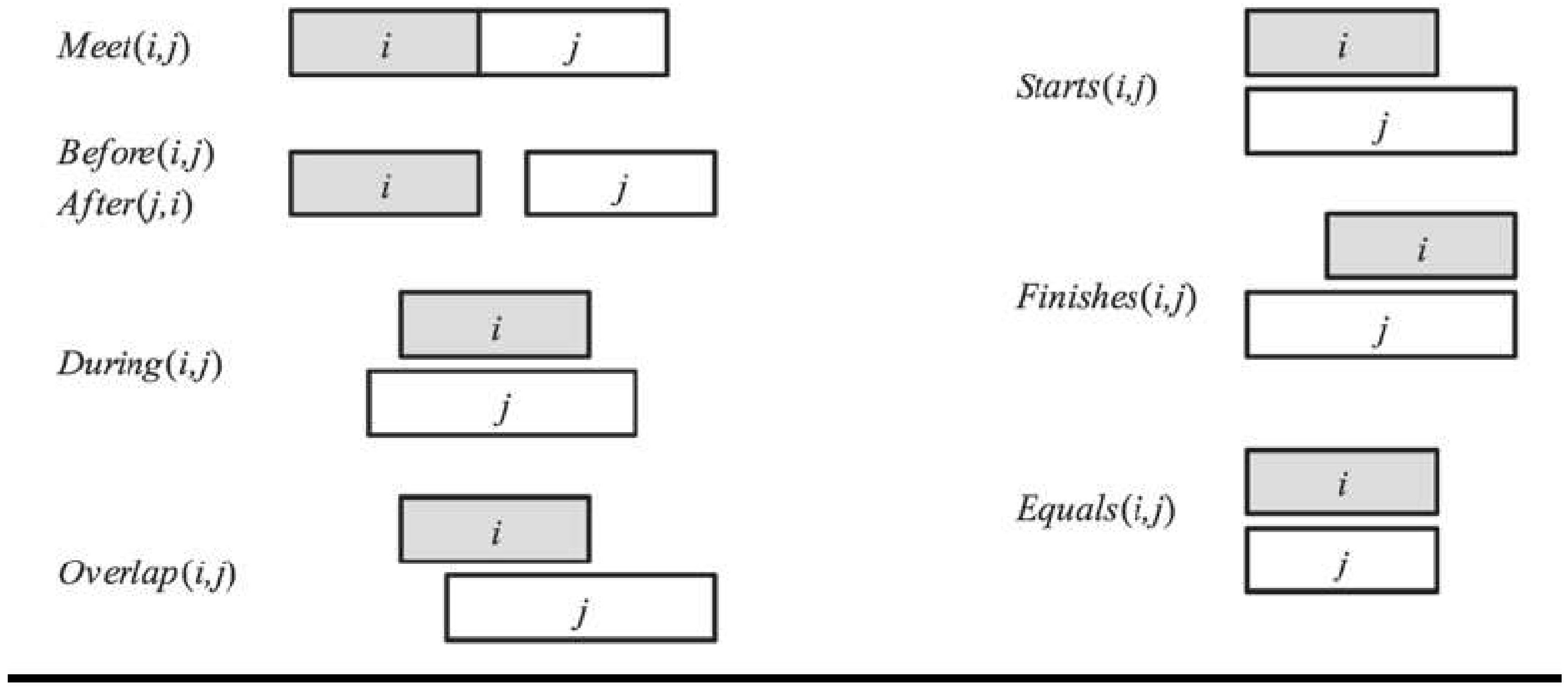
- If any event e that happens over an interval also happens at subinterval, then such categories of events are called process or liquid event categories.
- Any process e that happens over an interval also happens over any subinterval
- $(e \in Processes) \land Happens(e, (t1, t4)) \land (t1 < t2 < t3 < t4) \Rightarrow Happens(e, (t2, t3))$

Time intervals

- We will consider two kinds of time intervals: **moments** and **extended intervals**.
- The distinction is that only moments have zero duration:
- Partition({Moments, ExtendedIntervals}, Intervals)
- i ∈ Moments ⇔ Duration(i)=Seconds(0) .
- The functions **Begin** and **End** pick out the earliest and latest moments in an interval, and the function **Time** delivers the point on the time scale for a moment.(measure it in seconds, the moment at midnight (GMT) on January 1, 1900, has time 0)
- The function **Duration** gives the difference between the end time and the start time.
- Interval (i) ⇒ Duration(i)=(Time(End(i)) − Time(Begin(i))) .
- Time(Begin(AD1900))=Seconds(0) ;Time(Begin(AD2001))=Seconds(3187324800)
- Time(End(AD2001))=Seconds(3218860800) ;Duration(AD2001)=Seconds(31536000)

- a function Date, which takes six arguments (hours, minutes, seconds, day, month, and year) and returns a time point:
- Time(Begin(AD2001))=Date(0, 0, 0, 1, Jan, 2001)
- Date(0, 20, 21, 24, 1, 1995)=Seconds(3000000000) .
- The complete set of interval relations, as proposed by Allen (1983), is shown graphically in Figure and logically below:
- Meet(i, j) ⇔ End(i)=Begin(j)
- Before(i, j) ⇔ End(i) < Begin(j)
- After (j, i) ⇔ Before(i, j)
- During(i, j) ⇔ Begin(j) < Begin(i) < End(i) < End(j)
- Overlap(i, j) ⇔ Begin(i) < Begin(j) < End(i) < End(j)

Begins(i, j) ⇔ Begin(i) = Begin(j)
Finishes(i, j) ⇔ End(i) = End(j)
Equals(i, j) ⇔ Begin(i) = Begin(j) ∧ End(i) = End(j)



Predicates on time intervals.

Mental Events and Mental Objects

- Knowledge about one’s own knowledge and reasoning processes is useful for controlling inference.
- The agent should know what are in its knowledge base and what are not
- Knowledge about the knowledge of other agents is also important;
- What we need is a model of the mental objects that are in someone’s head (or something’s knowledge base) and of the mental processes that manipulate those mental objects.
- the **propositional attitudes** that an agent can have toward mental objects: attitudes such as Believes, Knows, Wants, Intends, and Informs.
- For eg. suppose we try to assert that Lois knows that Superman can fly:
- Knows(Lois, CanFly(Superman))

- if it is true that Superman is Clark Kent, then
- (Superman = Clark) ∧ Knows(Lois , CanFly(Superman))|= Knows(Lois, CanFly(Clark)) .
- if our agent knows that 2 + 2 = 4 and 4 < 5, then we want our agent to know that 2 + 2 < 5. This property is called **referential transparency**
- For propositional attitudes like *believes* and *knows*, we would like to have referential opacity
- **Modal logic** is designed to address this problem.
- Modal logic includes special modal operators that take sentences (rather than terms) as arguments.
- For example, "*A* knows *P*" is represented with the notation $\mathbf{K}_A P$, where **K** is the modal operator for knowledge

- we will need a more complicated model, one that consists of a collection of **possible worlds** rather than just one true world.
- The worlds are connected in a graph by **accessibility relations**, one relation for each modal operator.
- We say that world w1 is accessible from world w0 with respect to the modal operator $\mathbf{K}_A$ if everything in w1 is consistent with what A knows in w0, and we write this as Acc($\mathbf{K}_A$,w0,w1).
- In general, a knowledge atom $\mathbf{K}_A P$ is true in world w if and only if P is true in every world accessible from w.
- For example, we can say that, even though Lois doesn't know whether Superman's secret identity is Clark Kent, she does know that Clark knows:

$$\mathbf{K}_{Lois}[\mathbf{K}_{Clark} Identity(Superman, Clark) \vee \mathbf{K}_{Clark} \neg Identity(Superman, Clark)]$$

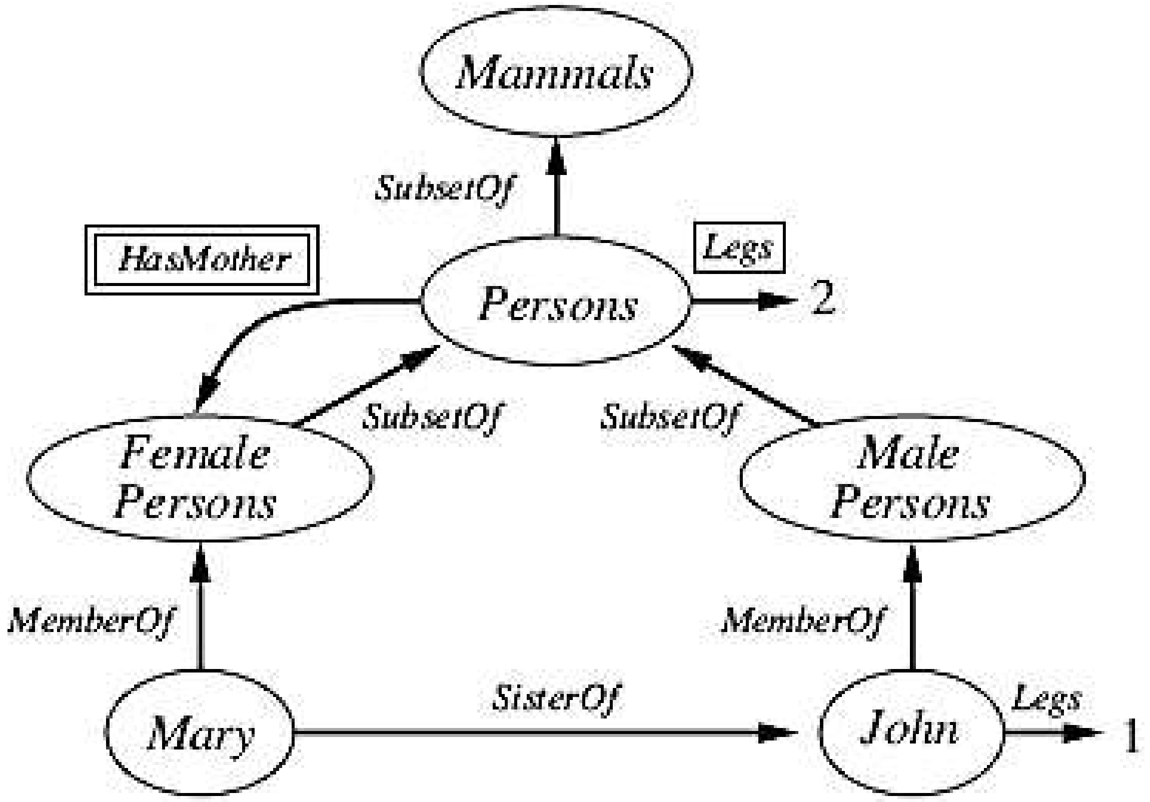
Reasoning Systems For Categories

- Categories are the primary building blocks of large-scale knowledge representation schemes.
- There are two systems specially designed for organizing and reasoning with categories.
- **semantic networks** provide graphical aids for visualizing a knowledge base and efficient algorithms for inferring properties of an object on the basis of its category membership;
- **description logics** provids a formal language for constructing and combining category definitions and efficient algorithms for deciding subset and superset relationships between categories.

Semantic Networks:

- a graphical notation of nodes and edges called **existential graphs.**
- There are many variants of semantic networks, but all are capable of representing individual objects, categories of objects, and relations among objects.
- A typical graphical notation displays object or category names in ovals or boxes, and connects them with labeled links
- For example, Figure has a MemberOf link between Mary and FemalePersons ,corresponding to the logical assertion

  Mary ∈ FemalePersons
- similarly, the SisterOf link between Mary and John corresponds to the assertion SisterOf (Mary, John).
- We can connect categories using SubsetOf links, and so on.

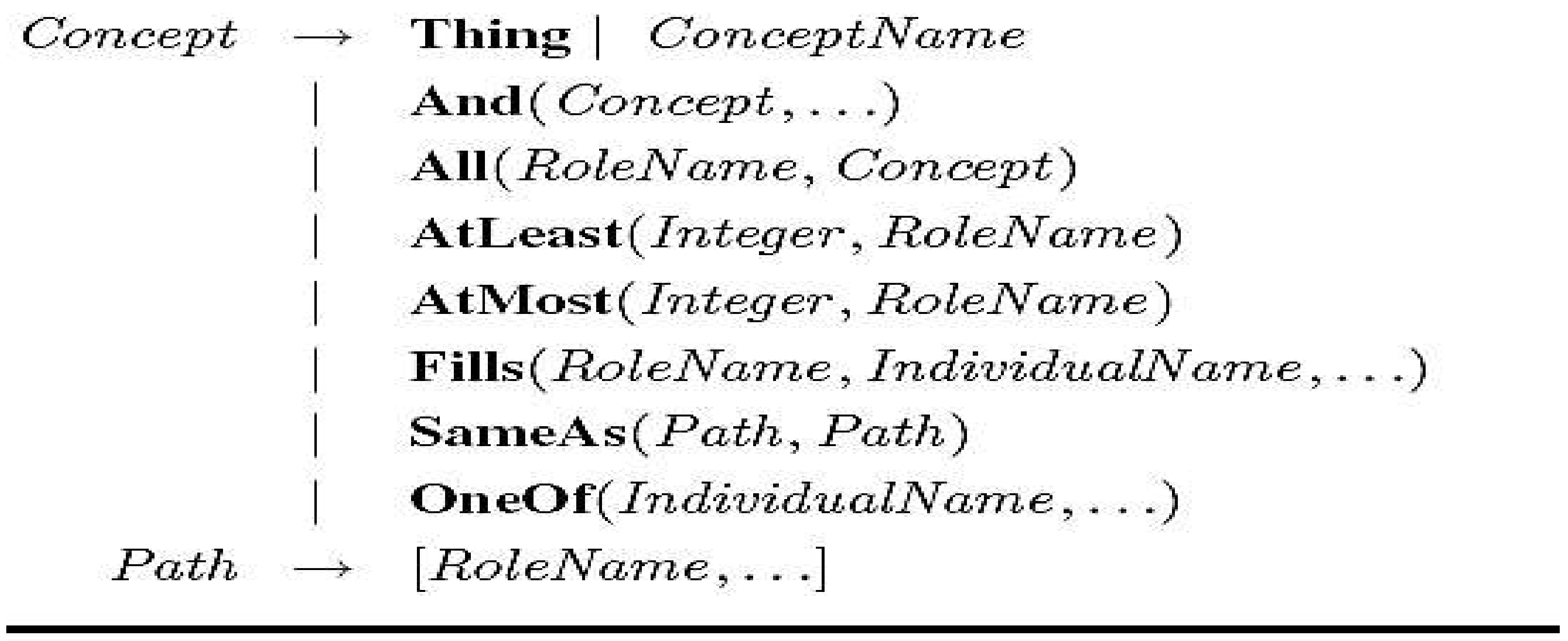


A semantic network with four objects (John, Mary, 1, and 2) and four categories. Relations are denoted by labeled links.

- we have used a special notation—the double-boxed link—in Figure
- $\forall x\ x \in Persons \Rightarrow [\forall\ y\ HasMother\ (x, y) \Rightarrow y \in FemalePersons\ ]$
- We might also want to assert that persons have two legs—that is,
- $\forall x, x \in Persons \Rightarrow Legs(x, 2)$.
- the single-boxed link in Figure is used to assert properties of every member of a category.
- The semantic network notation makes it convenient to perform **inheritance** reasoning

Description logics

- **Description logics** are notations that are designed to make it easier to describe definitions and properties of categories.
- The principal inference tasks for description logics are **subsumption** (checking if one category is a subset of another by comparing their definitions) and **classification** (checking whether an object belongs to a category)
- Some systems also include **consistency** of a category definition—whether the membership criteria are logically satisfiable
- The CLASSIC language is a typical description logic. The syntax of CLASSIC descriptions is shown in Figure



$$
\begin{array}{lcl}
\mathit{Concept} & \rightarrow & \textbf{Thing} \mid \mathit{ConceptName} \\
 & \mid & \textbf{And}(\mathit{Concept}, \ldots) \\
 & \mid & \textbf{All}(\mathit{RoleName}, \mathit{Concept}) \\
 & \mid & \textbf{AtLeast}(\mathit{Integer}, \mathit{RoleName}) \\
 & \mid & \textbf{AtMost}(\mathit{Integer}, \mathit{RoleName}) \\
 & \mid & \textbf{Fills}(\mathit{RoleName}, \mathit{IndividualName}, \ldots) \\
 & \mid & \textbf{SameAs}(\mathit{Path}, \mathit{Path}) \\
 & \mid & \textbf{OneOf}(\mathit{IndividualName}, \ldots) \\
\mathit{Path} & \rightarrow & [\mathit{RoleName}, \ldots]
\end{array}
$$

The syntax of descriptions in a subset of the CLASSIC language.

For example, to say that bachelors are unmarried adult males we would write

Bachelor = And(Unmarried, Adult ,Male) .

The equivalent in first-order logic would be

Bachelor (x) ⇔ Unmarried(x) ∧ Adult(x) ∧ Male(x) .

- Any description in CLASSIC can be translated into an equivalent first-order sentence, but some descriptions are more straightforward in CLASSIC
- For example, to describe the set of men with at least three sons who are all unemployed and married to doctors, and at most two daughters who are all professors in physics or math departments, we would use

```
And(Man, AtLeast(3, Son), AtMost(2, Daughter ),
All(Son, And(Unemployed,Married, All(Spouse, Doctor ))),
All(Daughter , And(Professor , Fills(Department , Physics,Math))))
```

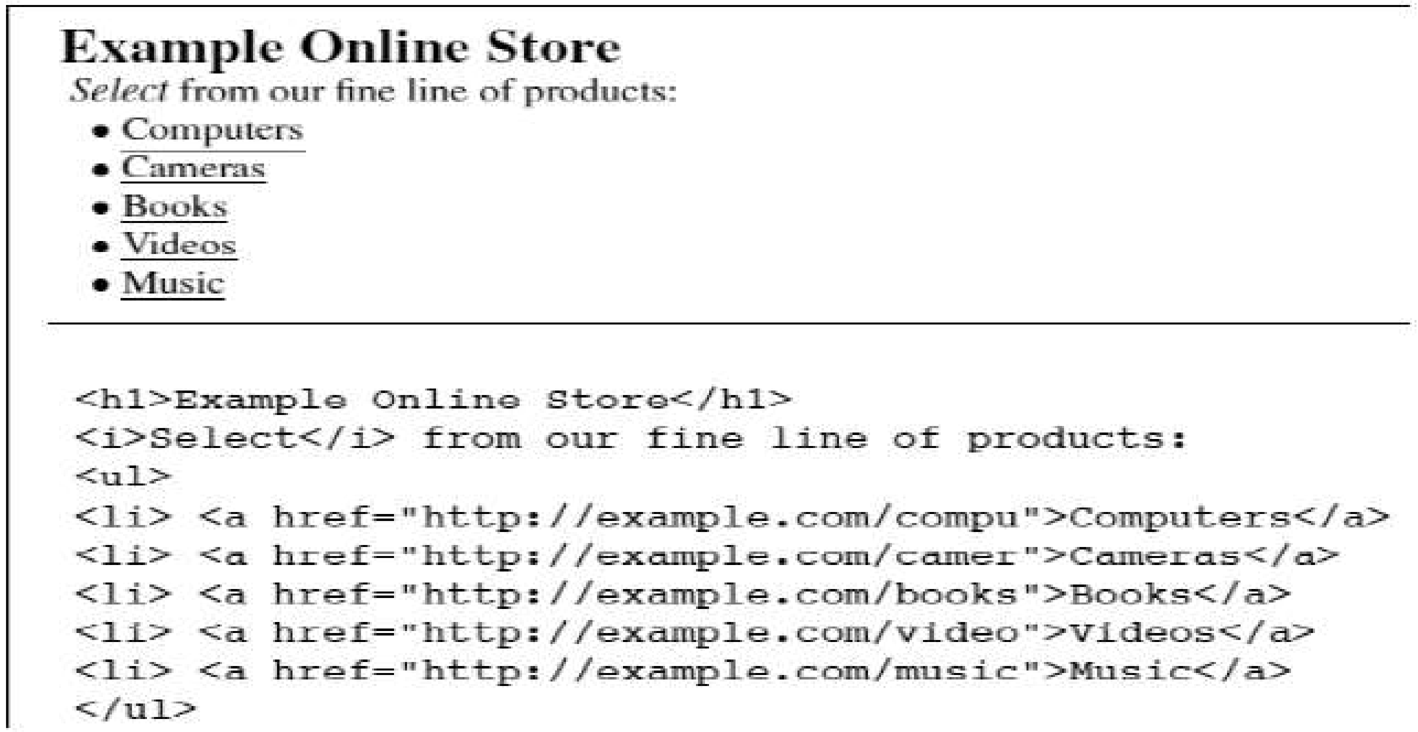
REASONING WITH DEFAULT INFORMATION

- **Circumscription** can be seen as a more powerful and precise version of the closed world assumption.
- The idea is to specify particular predicates that are assumed to be “as false as possible”—that is, false for every object except those for which they are known to be true.
- For example, suppose we want to assert the default rule that birds fly.
- We would introduce a predicate, say Abnormal1(x), and write
- $Bird(x) \land \neg Abnormal1(x) \Rightarrow Flies(x)$.
- If we say that Abnormal 1 is to be **circumscribed**, a circumscriptive reasoner is entitled to assume ¬Abnormal 1(x) unless Abnormal 1(x) is known to be true.
- This allows the conclusion Flies(Tweety) to be drawn from the premise Bird(Tweety), but the conclusion no longer holds if Abnormal1(Tweety) is asserted.

- **Default logic** is a formalism in which **default rules** can be written to generate contingent, nonmonotonic conclusions. A default rule looks like this:
- Bird(x) : Flies(x)/Flies(x) .
- This rule means that if Bird(x) is true, and if Flies(x) is consistent with the knowledge base, then Flies(x) may be concluded by default.
- In general, a default rule has the form
- P : J1, . . . , Jn/C
- where P is called the prerequisite, C is the conclusion, and Ji are the justifications.
- if any one of them can be proven false, then the conclusion cannot be drawn.
- Any variable that appears in Ji or C must also appear in P

The Internet Shopping world:

- In this section , we will create a shopping research agent that helps a buyer find product offers on the Internet .
- The shopping agent is given a product description by the buyer and has the task of producing a list of Web pages that offer such a product for sale and ranking for the best.



Example Online Store

Select from our fine line of products:

- Computers
- Cameras
- Books
- Videos
- Music

```
<h1>Example Online Store</h1>
<i>Select</i> from our fine line of products:
<ul>
<li> <a href="http://example.com/compu">Computers</a>
<li> <a href="http://example.com/camer">Cameras</a>
<li> <a href="http://example.com/books">Books</a>
<li> <a href="http://example.com/video">Videos</a>
<li> <a href="http://example.com/music">Music</a>
</ul>
```

The above figure shows a web page and a corresponding HTML character string.

- The Agents first task is to find relevant product offers.
- Let us consider query be a product description that the user types in (e.g.”laptop”);
- then a page is relevant offer for query,if the page is relevant and the page is indeed an offer. Also keep track of URL associated with the page.
- RelevantOffer(page,url,query) ⇔Relevant(page,url,query)^Offer(page).
- We can say a page is an offer if it contains the word ”buy” or “price” within an HTML link or form on the page.
- If the page contains a string of the form “<a...buy...</a>” then it is an offer, it could also be say “price” instead of “buy” or use “form” instead of “a”. We can write axioms for this:
- Offer(page) ⇔ (InTag(“a”,str,page)v InTag(“form”,str,page)) ^

  (In(“buy”,str) v In(“price”,str)).

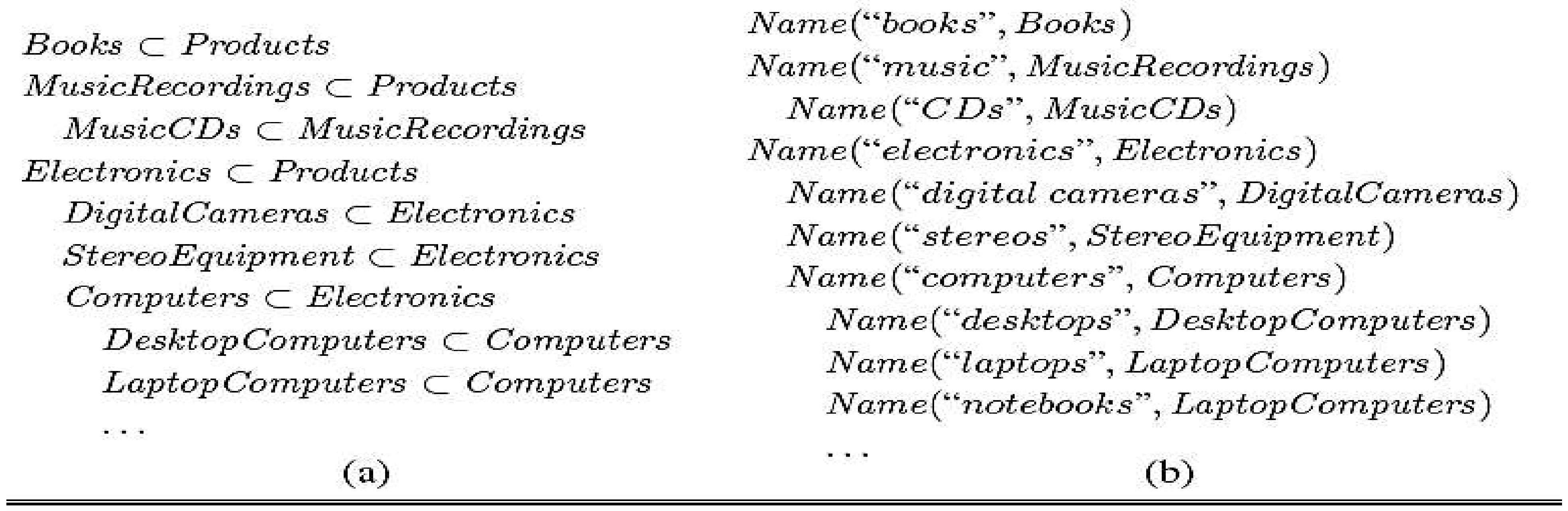
- We need to find the relevant pages.
- The strategy is to start at the home page of an online store and consider all the pages that can be reached by the following relevant links.
- The Agent will have a knowledge of a number of stores, for example:

  Amazon $\in$OnlineStores $\wedge$ Homepage(Amazon, “amazon.com”) .

  Ebay $\in$OnlineStores $\wedge$ Homepage(Ebay, “ebay.com”) .

  ExampleStore $\in$OnlineStores $\wedge$ Homepage(ExampleStore, “example.com”)
- These stores classify their goods into product categories, and provide links to the Major categories from their home page.
- Minor categories can be reached by following a chain of relevant links, and eventually we will reach offers .
- A page is relevant to the query if it can be reached by a chain of relevant category links from a stores home page, and following one more link to the product offer :

- Relevant(page, query) ⇔ ∃ store, home store ∈OnlineStores ∧ Homepage(store, home) ∧ ∃url , url 2 RelevantChain(home, url 2, query) ∧ Link(url 2, url) ∧ page = Contents(url)
- A chain of links between two URLs, start and end ,is relevant to a description d if the anchor text of each link is relevant category name for d.
- The existence of the chain itself is determined by a recursive definition , with the empty chain (start = end)as the base case:
- First we need to relate strings to the categories they name.
- This is done by using the predicate Name(s,c), which says that string s is a name for category c –for example , we might assert that Name(“laptops”,LaptopComputers).



$Books \subset Products$
$MusicRecordings \subset Products$
$\quad MusicCDs \subset MusicRecordings$
$Electronics \subset Products$
$\quad DigitalCameras \subset Electronics$
$\quad StereoEquipment \subset Electronics$
$\quad Computers \subset Electronics$
$\quad\quad DesktopComputers \subset Computers$
$\quad\quad LaptopComputers \subset Computers$
$\quad\quad \ldots$

(a)

$Name(\text{“books”}, Books)$
$Name(\text{“music”}, MusicRecordings)$
$\quad Name(\text{“CDs”}, MusicCDs)$
$Name(\text{“electronics”}, Electronics)$
$\quad Name(\text{“digital cameras”}, DigitalCameras)$
$\quad Name(\text{“stereos”}, StereoEquipment)$
$\quad Name(\text{“computers”}, Computers)$
$\quad\quad Name(\text{“desktops”}, DesktopComputers)$
$\quad\quad Name(\text{“laptops”}, LaptopComputers)$
$\quad\quad Name(\text{“notebooks”}, LaptopComputers)$
$\quad\quad \ldots$

(b)

Figure 12.9 (a) Taxonomy of product categories. (b) Names for those categories.

- Suppose the query is “laptops” , then RelevantCategoryName(query,text) is true when one of the following holds:
- The text and query name the same category—e.g.,”laptop computers” and “laptops”.
- The text names a super category such as “computers”.
- The text names a subcategory such as “ultralight notebooks”.
- The logical definition of RelevantCategoryName is as follows:
- RelevantCategoryName(query, text) ⇔ ∃ c1, c2 Name(query, c1) ∧ Name(text, c2) ∧ (c1 ⊆ c2 ∨ c2 ⊆ c1)

Comparing offers:

- To compare offers ,the agent must extract the relevant information –price ,speed ,disk size ,weight, and so on—from the offer pages.
- This can be difficult task with real web pages. A common way of dealing with this problem is to use programs called wrappers to extract information from a page.
- Consider given a page on the gen-store.com site with the text
- YVM ThinkBook 970. Our price: $1449.00
- Followed by various technical specifications , we would like a wrapper to extract information such as the following:
- ∃ c, offer c∈ LaptopComputers ∧ offer ∈ ProductOffers ∧ Manufacturer(c, IBM) ∧ Model (c, ThinkBook970) ∧ ScreenSize(c, Inches(14)) ∧ ScreenType (c, ColorLCD) ∧ MemorySize(c,Gigabytes(2)) ∧ CPUSpeed (c,GHz (1.2)) ∧ OfferedProduct(offer, c) ∧ Store(offer , GenStore) ∧ URL(offer , “example.com/computers/34356.html”) ∧ Price(offer , $(399)) ∧ Date(offer ,Today)

- The final task is to compare the offers that have been extracted . For example , consider these three offers:

  A : 1.4 GHz CPU, 2GB RAM, 250 GB disk, $299 .

  B : 1.2 GHz CPU, 4GB RAM, 350 GB disk, $500 .

  C : 1.2 GHz CPU, 2GB RAM, 250 GB disk, $399 .
- C is dominated by A; that is , A is cheaper and faster ,and they are otherwise the same.
- The shopping agent we have described here is a simple one ; many refinements are possible.

Unit-5

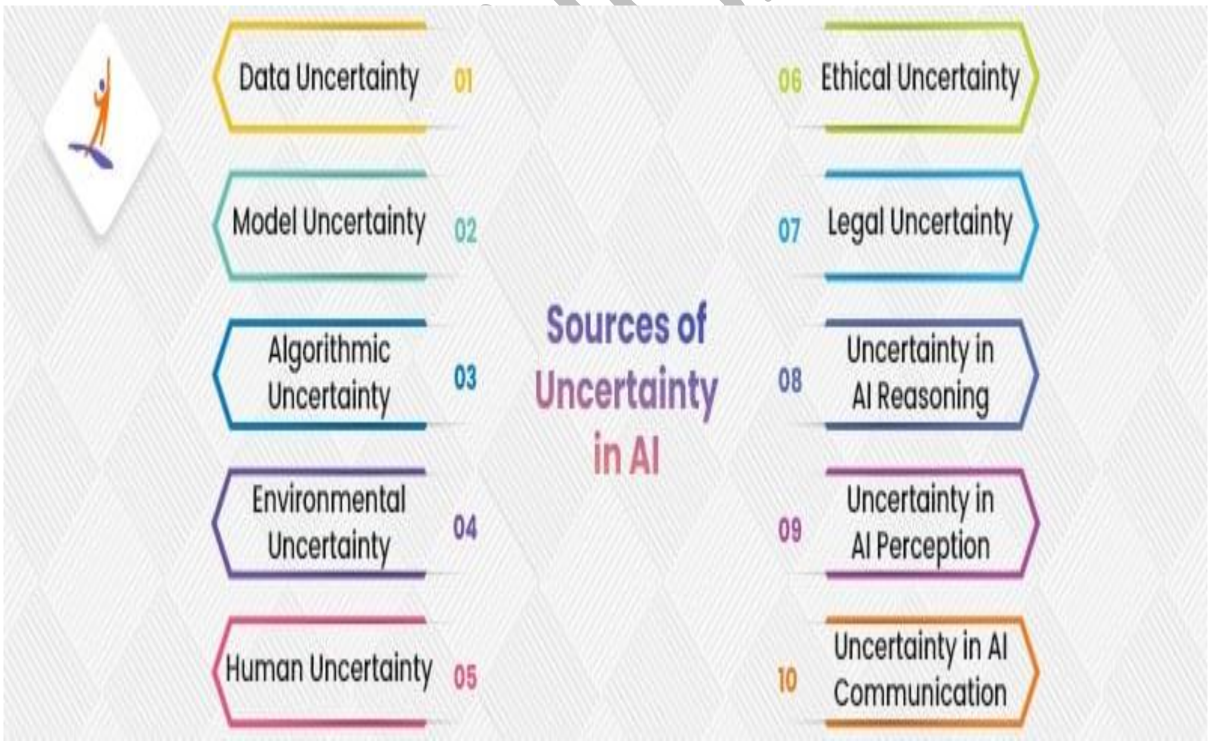
Quantifying Uncertainty

Acting Under Uncertainty

- Artificial intelligence (AI) uncertainty is when there's not enough information or ambiguity in data or decision-making. It is a fundamental concept in AI, as real-world data is often noisy and incomplete. AI systems must account for uncertainty to make informed decisions.
- AI deals with uncertainty by using models and methods that assign probabilities to different outcomes. Managing uncertainty is important for AI applications like self-driving cars and medical diagnosis, where safety and accuracy are key

Sources of Uncertainty in AI

There are several sources of uncertainty in AI that can impact the reliability and effectiveness of AI systems. Here are some common sources of uncertainty in AI:


Data Uncertainty 01
Model Uncertainty 02
Algorithmic Uncertainty 03
Environmental Uncertainty 04
Human Uncertainty 05
Sources of Uncertainty in AI
06 Ethical Uncertainty
07 Legal Uncertainty
08 Uncertainty in AI Reasoning
09 Uncertainty in AI Perception
10 Uncertainty in AI Communication

Data Uncertainty: AI models are trained on data, and the quality and accuracy of the data can affect the performance of the model. Noisy or incomplete data can lead to uncertain predictions or decisions made by the AI system.

Model Uncertainty: AI models are complex and can have various parameters and hyperparameters that need to be tuned. The choice of model architecture, optimization algorithm, and hyperparameters can significantly impact the performance of the model, leading to uncertainty in the results.

Algorithmic Uncertainty: AI algorithms can be based on different mathematical formulations, leading to different results for the same problem. For example, different machine learning algorithms can produce different predictions for the same dataset.

Environmental Uncertainty: AI systems operate in dynamic environments, and changes in the environment can affect the performance of the system. For example, an autonomous vehicle may encounter unexpected weather conditions or road construction that can impact its ability to navigate safely.

Human Uncertainty: AI systems often interact with humans, either as users or as part of the decision-making process. Human behaviour and preferences can be difficult to predict, leading to uncertainty in the use and adoption of AI systems.

Ethical Uncertainty: AI systems often raise ethical concerns, such as privacy, bias, and transparency. These concerns can lead to uncertainty in the development and deployment of AI systems, particularly in regulated industries.

Legal Uncertainty: AI systems must comply with laws and regulations, which can be ambiguous or unclear. Legal challenges and disputes can arise from the use of AI systems, leading to uncertainty in their adoption and implementation.

Uncertainty in AI Reasoning: AI systems use reasoning techniques to make decisions or predictions. However, these reasoning techniques can be uncertain due to the complexity of the problems they address or the limitations of the data used to train the models.

Uncertainty in AI Perception: AI systems perceive their environment through sensors and cameras, which can be subject to noise, occlusion, or other forms of interference. This can lead to uncertainty in the accuracy of the data used to train AI models or the effectiveness of AI systems in real-world applications.

Uncertainty in AI Communication: AI systems communicate with humans through natural language processing or computer vision. However, language and visual cues can be ambiguous or misunderstood, leading to uncertainty in the effective communication between humans and AI systems.

Types of Uncertainty in AI

Aleatoric Uncertainty: This type of uncertainty arises from the inherent randomness or variability in data. It is often referred to as "data uncertainty." For example, in a classification task, aleatoric uncertainty may arise from variations in sensor measurements or noisy labels.

Epistemic Uncertainty: Epistemic uncertainty is related to the lack of knowledge or information about a model. It represents uncertainty that can potentially be reduced with more data or better modelling techniques. It is also known as "model uncertainty" and arises from model limitations, such as simplifications or assumptions.

Parameter Uncertainty: This type of uncertainty is specific to probabilistic models, such as Bayesian neural networks. It reflects uncertainty about the values of model parameters and is characterized by probability distributions over those parameters.

Uncertainty in Decision-Making: Uncertainty in AI systems can affect the decision-making process. For instance, in reinforcement learning, agents often need to make decisions in environments with uncertain outcomes, leading to decision-making uncertainty.

Uncertainty in Natural Language Understanding: In natural language processing (NLP), understanding and generating human language can be inherently uncertain due to language ambiguity, polysemy (multiple meanings), and context-dependent interpretations.

Probability Notation

- Probabilistic notation refers to the symbols and conventions used to represent and manipulate probabilities and statistical concepts.
- This notation is fundamental in fields such as statistics, machine learning, and artificial intelligence

Basic Probabilistic Notations

Here are some key elements of probabilistic notation, which form the foundation for more advanced probabilistic models in AI:

Probability Notation:

Probability Notation	**Description**
$P(A)$	The probability of event A occurring
$P(A')$	The probability of event A not occurring
$P(A \cap B)$	The probability of both A and B occurring at the same time
$P(A \cup B)$	The probability of either A or B occurring
$P(A \cap B')$	The probability of A occurring but not B
$P(A' \cup B)$	The probability of either A not occurring or B occurring

Conditional Probability:

- **P(A | B)**: The probability of event A occurring given that event B has occurred. This is fundamental in AI for updating beliefs based on new evidence.
- **Bayes' Theorem**: $P(A|B)=P(B)P(B|A)\cdot P(A)P(A|B)=P(B)P(B|A)\cdot P(A)$, which provides a way to update probabilities based on new data.

Joint Probability:

- The probability of both A and B occurring, which can also be written as $P(A \cap B)$. This is essential for understanding the relationships between multiple variables.

Marginal Probability:

- The probability of event A **P(A)** occurring, regardless of other events. This is derived by summing or integrating over the joint probabilities of A with all other possible events.

Advanced Probabilistic Notations

Random Variables:

- **X**: A random variable representing a possible outcome.

- **P(X = x)**: The probability that the random variable X takes the value x.
- **P(X ≤ x)**: The probability that the random variable X takes a value less than or equal to x.

Probability Distributions:

- **Probability Mass Function (PMF)**: For discrete random variables, $P(X=x)P(X=x)$denotes the PMF.
- **Probability Density Function (PDF)**: For continuous random variables, $fX(x)fX(x)$ denotes the PDF.
- **Cumulative Distribution Function (CDF)**: $FX(x)=P(X\leq x)FX(x)=P(X\leq x)$ gives the cumulative probability up to x.

Expectation and Variance:

- **E[X]**: The expected value or mean of the random variable X.
- **Var(X)**: The variance of the random variable X, representing the spread of its possible values.

Covariance and Correlation:

- **Cov(X, Y):** The covariance between random variables X and Y, indicating the degree to which they change together.
- **Corr(X, Y)**: The correlation coefficient between X and Y, a normalized measure of their linear relationship.

Inference Using Full Joint Distributions

- Joint probability offers valuable insights into the likelihood of multiple events happening together. This helps us in several ways:

Co-occurrence: Joint probability helps us understand how likely it is for two or more events to happen at the same time. This is important for seeing how events are connected and the probability of them occurring together.

Risk Evaluation: In areas like finance and insurance, joint probability helps us assess the risk when multiple events overlap. For instance, it can estimate the chance of multiple financial instruments facing losses simultaneously.

Quality Check: Businesses can use joint probability to gauge the reliability and quality of their products or processes. It shows the likelihood of multiple defects or issues occurring at once, which allows for proactive quality improvement efforts.

Event Relationships: Joint probability can indicate if events are related or not. If joint probability significantly differs from the product of individual probabilities, it suggests events are connected, and the occurrence of one affects the likelihood of the other.

Decision Support: When businesses need to make choices involving multiple factors or events, joint probability provides a numerical foundation for decision-making. It helps assess how different variables together impact the desired outcome.

Resource Management: In situations with limited resources, understanding joint probability helps optimise resource allocation. For example, in supply chain management, it can estimate the chance of multiple supply chain disruptions happening at the same time, enabling better risk management strategies.

Formula for Joint Probability

For Independent Events

When events A and B are independent, meaning that the occurrence of one event does not impact the other, we use the multiplication rule:

$P(A \cap B) = P(A) \times P(B)$

Here, P(A) is the probability of occurrence of event A, P(B) is the probability of occurrence of event B, and $P(A \cap B)$ is the joint probability of events A and B.

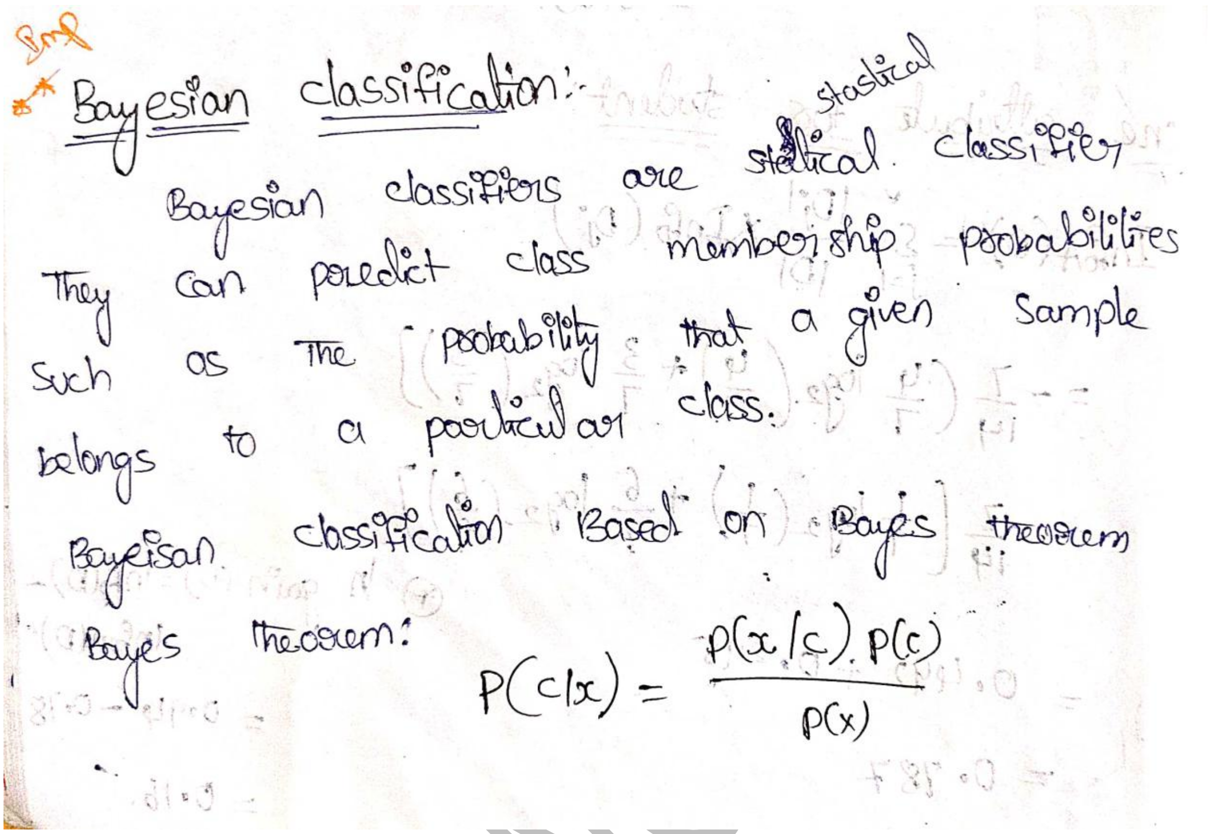
For Dependent Events

Events are often dependent on each other, meaning that one event's occurrence influences the likelihood of the other. Here, we employ a modified formula:

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

Here, P(A) is the probability of occurrence of event A, P(B|A) is the conditional probability of occurrence of event B when event A has already occurred, and $P(A \cap B)$ is the joint probability of events A and B.

Bayesian Classification:



Imp

* Bayesian classification:-

Bayesian classifiers are stastical ~~statical~~ classifier

They can predict class membership probabilities such as the probability that a given sample belongs to a particular class.

Bayeisan classification Based on Bayes theorem

Baye's theorem:

$$P(c|x) = \frac{p(x/c).p(c)}{p(x)}$$

It finds postperior probability and posibrity probabili-ty.

* It find the postprior probability of a class conditinal.

Algorithm:-

step 1:-

* let a 'D' be a training dataset of Data tuple associated with class lables.

step 2:-

* each tuple is represented by an N-dimensional vector $x = (x_1, x_2 ---x_n)$ where $x_1, x_2 ---x_n$ are values of attributes.

* Suppose that there are m no. of classes $C_1, C_2 ---C_m$ given a data tuple x the classifier is predict that $x \in$ the class having the highest posterior probability condition on x.

* This can be written mathematically where $j = 1, 2, ---m$ and $i \neq j$.

i.e, we maximize $P(C_i|x)$

Bayes theorem $P(C_i|x) = \frac{P(x|C_i)P(C_i)}{P(x)}$

step 3:-

As $P(x)$ is constant for all classes it is enough to maximize only that is numerical function has $P(x|C_i)$.

* If the class probability is not known

$p(C_1) = p(C_2) = \cdots = p(C_m)$

so it is enough to maximize only $p(x/C_i)$

step 4:-

$$p(x/C_i) = \prod_{k=1}^{m} p(x_k | C_i)$$

$$= p(x_1/C_i) \times p(x_2|C_i) \times \cdots \times p(x_m|C_i)$$

(i) If A_k categorical $p(x_k|C_i)$ is number of tuples of class C_i in 'D' having the value small S_k for A_k / no. of tuples of C_i in D.

(ii) A_k is continuous $g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\,\sigma_{C_i}} \times e^{\frac{-(x-\mu)^2}{2\sigma^2}}$

where $g(x, \mu, \sigma)$ is the Gaussian (normal) density function for attribute A_k while μ and σ are the mean and standard deviation, respectively given the values for attributes A_k for training samples of class C.

step 5:-

The classifier predict the class lable is C_i if and only if $p(x/C_i)\,p(C_i) > p(x|C_j) \cdot p(C_j)$ for $1 \le j \le m$, $j \ne i$

Problem :-

RID	Age	Income	student	credit rating	buys Computer.
1	youth	high	NO	Avg	NO
2	"	"	"	Exce.	"
3	middle	"	"	Avg	yes
4	senior	medium	"	"	"
5	senior	low	yes	"	"
6	senior	"	yes	Exce	NO
7	middle	"	yes	"	yes
8	youth	medium	~~yes~~ NO	Avg	NO
9	youth	low	~~NO~~ yes	"	yes
10	senior	medium	yes	"	"
11	youth	"	"	Exce	"
12	middle	"	NO	Exce	"
13	middle	high	yes	Avg	yes
14	senior	medium	NO	Exce	NO

X = (age = "<= 30", income = "medium", student = "yes", credit - rating = "Fair" avg.) predict the class labeled data tuple 'x' using bayesian classification algorithm.

Sol :- The prior Probability of each class.

P(buys_computer = "yes") = $\frac{9}{14}$ = 0.6428

P(buys - computer = "NO") = $\frac{5}{14}$ = 0.3571

young:-

P(age <= 30 | buys . computer = "yes") = $\frac{2}{9}$ = 0.222

P(age <= 30 | buys computer = "NO") = $\frac{3}{5}$ = 0.600

medium:-

P(income = "medium" | buys computer = "yes") = $\frac{4}{9}$ = 0.444

P(income = "medium" | buys computer = "NO") = $\frac{2}{5}$ = 0.40

student:-

P(student = "yes" | buys computer = "yes") = $\frac{6}{9}$ = 0.666

P(student = "yes" | buys computer = "NO") = $\frac{1}{5}$ = 0.200

credit - rating:-

P(credit_rating = "avg" | buys computer = "yes") = $\frac{6}{9}$ = 0.666

P(" = "avg" | buys computer = "NO") = $\frac{2}{5}$ = 0.400

using the above probabilities:-

P(x | buys.computer = "yes") = 0.222 × 0.444 × 0.666 × 0.666

= 0.0437

P(x | buys. computer = "NO") = 0.600 × 0.400 × 0.200 × 0.400

= 0.019