AIMA CHAPTER 12: KNOWLEDGE REPRESENTATION

Adapted from slides by Macskassy and slides by M.E. Latoschik

GOAL OF CHAPTER 12: KNOWLEDGE REPRESENTATION

- Previously we learned about
  - Logical Agents
  - Planning with Logical Agents
- What contents to put into the Knowledge Base.
  - How to represent facts about the world.
- Content:
  - General ontology
  - Categories and objects
  - Events and processes
  - Reasoning systems
  - Example> Internet shopping world

GOOD KNOWLEDGE BASE

- Referential Uniqueness
- Make References Explicit
- Semantic Uniqueness
- Functional Uniqueness
- Expressive
- Concise
- Unambiguous
- Context-insensitive
- Effective
- Clear
- Correct

CONSIDERATIONS

- The book & slide will use first-order logic (FOL) to discuss the content and organization of knowledge
  - Certain aspects of the real world are hard to capture in FOL.
    - Most generalizations have exceptions or hold only to a degree.
    - For example, although "tomatoes are red" is a useful rule, some tomatoes are green, yellow, or orange.
    - Similar exceptions can be found to almost all the rules in this chapter.
- The ability to handle exceptions and uncertainty is extremely important, but is orthogonal to the task of understanding the general ontology.

PROPOSITIONAL LOGIC VS FOL

Propositional logic

- **Logical constants:** true, false
- **Propositional symbols:** P, Q, S, ... (**atomic sentences**)
- Wrapping **parentheses:** (...)
- Sentences are combined by **connectives:**
  - ∧ ...and [conjunction]
  - ∨ ...or [disjunction]
  - ⇒...implies [implication]
  - ⇔..is equivalent [biconditional]
  - ¬ ...not [negation]
- **Literal:** atomic sentence or negated atomic sentence

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"

FIRST-ORDER LOGIC CONT.

- **User provides:**
  - **Constant symbols,** which represent individuals in the world
  - **Function symbols,** which map individuals to individuals
  - **Predicate symbols,** which map individuals to truth values
- FOL Provides
  - Variable symbols
    - E.g., x, y, foo
  - Connectives
    - Same as in PL: not (¬), and (∧), or (∨), implies (→), if and only if (biconditional ↔)
  - Quantifiers
    - Universal ∀x or **(Ax)**
    - Existential ∃x or **(Ex)**

- Sentences are built from terms and atoms
  - A **term** 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:
    - ¬P, P∨Q, P∧Q, P→Q, P↔Q where P and Q are sentences
  - A **quantified sentence** adds quantifiers ∀ and ∃
  - A **well-formed formula (wff)** is a sentence containing no "free" variables. That is, all variables are "bound" by universal or existential quantifiers.
    - (∀x)P(x,y) has x bound as a universally quantified variable, but y is free.

ONTOLOGICAL ENGINEERING

- Representation of abstracts concepts, such as *Events, Time, Physical Objects, and Beliefs* is called the **Ontological Engineering.**

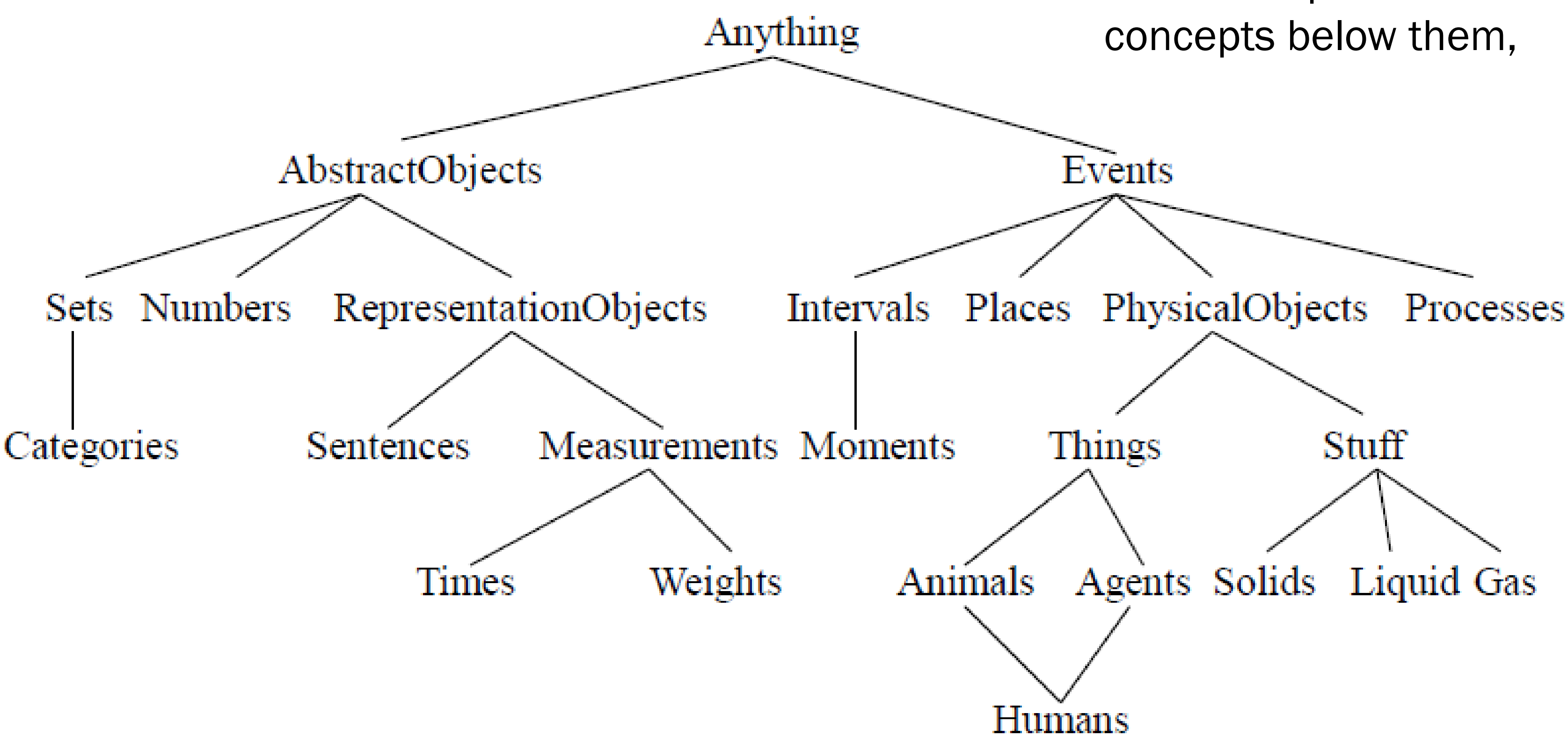
- **Ontology:** Collection of concepts and inter-relationships
  - Widely used in the database community to "translate" queries and concepts from one database to another, so that multiple databases can be used conjointly

- **Knowledge Engineer:** Populates KB with facts and relations

THE UPPER ONTOLOGY

- General framework of concepts

General framework of concepts is called an **upper ontology** because of the convention of drawing graphs with the general concepts at the top and the more specific concepts below them,



GENERAL ONTOLOGICAL ENGINEERING

- A general-purpose ontology should be applicable in more or less any special-purpose domain (with the addition of domain-specific axioms).
- In any sufficiently demanding domain, different areas of knowledge must be *unified*,
  - Reasoning and problem solving could involve several areas simultaneously.
- As Tom Gruber (2004) says,
  - "Every ontology is a treaty—a social agreement—among peo ple with some common motive in sharing."

GENERAL ONTOLOGICAL ENGINEERING

- Enterprise of general ontological engineering has so far had onl y limited success.
- Those ontologies that do exist have been created along four rou tes:
  - By a team of trained ontologist/logicians, who architect the ontology and write axioms.
    - The CYC system was mostly built this way (Lenat and Guha, 1990).
  - By importing categories, attributes, and values from an existing databas e or databases.
    - DBPEDIA was built by importing structured facts from Wikipedia (Bizer *et al.*, 200 7).
  - By parsing text documents and extracting information from them.
    - TEXTRUNNER was built by reading a large corpus of Web pages (Banko and Etzio ni, 2008).
  - By enticing unskilled amateurs to enter commonsense knowledge.
    - The OPENMIND system was built by volunteers who proposed facts in English (Si ngh *et al.*, 2002; Chklovski and Gil, 2005).

GENE ONTOLOGY

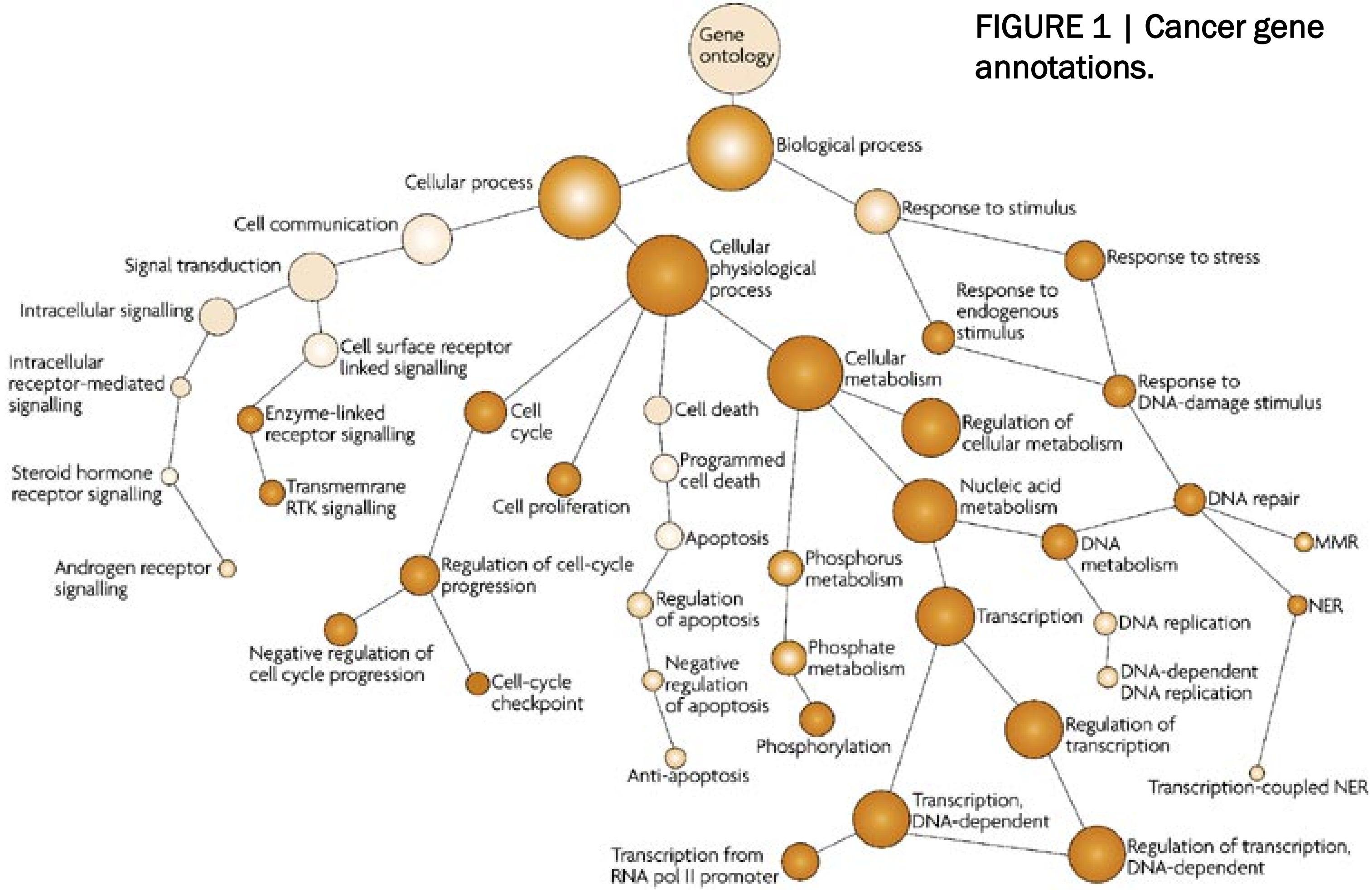


FIGURE 1 | Cancer gene annotations.

FROM THE FOLLOWING ARTICLE:
[Computational prediction of cancer-gene function]
Pingzhao Hu, Gary Bader, Dennis A. Wigle & Andrew Emili
Nature Reviews Cancer **7**, 23-34 (January 2007)
doi:10.1038/nrc2036

TOWARDS A GENERAL ONTOLOGY

- Develop good representations for:
  - categories
  - measures
  - composite objects
  - time, space and change
  - events and processes
  - physical objects
  - substances
  - mental objects and beliefs

CATEGORIES AND OBJECTS

- We interact with individual objects, but much of reasoning takes place at the level of **categories**.
  - The organization of objects into categories is a vital part of KR.
- Categories are also needed to make predictions about obj ects once they are classified.
  - Infers the presence of certain objects from perceptual input,
  - Infers category membership from the perceived properties of the objects, and then
  - Uses category information to make predictions about the objects
- Important relationships are

  subclass relation (AKO - a kind of)

  <category> AKO <category>.

  instance relation (ISA - is a)

  <object> ISA <category>.

CATEGORY REPRESENTATIONS

- There are two choices of representing categories in first order logic: predicates and objects. That is, we can use the predicate Basketball(b) or we can **reify** the category as an "object" *basketball*. We could then write

  member(x,basketball) or

  $x \in$ basketball

- We will also use the notation

  isa(x,basketball).

- Basketball is a subset or subcategory of Ball, which is abbreviated

  Basketball $\subset$ Ball

- We will also use the notation

  ako(basketball,ball).

*__reification:__ turn a predicate or function into an object

REIFYING PROPERTIES

- An individual object may have a property.
- For example, a specific ball, BB9 can be round.
- In ordinary FOL, we write

  Round(BB9).
- As for categories, we can <u>regard Round as higher order object</u>, and say

  BB9 has the property Round
- We will also use the notation

  hasprop(BB9,round).

REIFYING PROPERTY VALUES

- Some properties are determined by an attribute and a value. For example, the diameter of my basketball BB9 has diameter 9.5:

  Diameter(BB9)=9.5

- We can also use the notation

  has(bb9,diameter,9.5).

- An alternative representation for properties , when regarded as Boolean attributes is

  has(BB9,round,true).

- In the same manner, we can express that a red ball has colour = red.

  has(BB9,colour,red).

TAXONOMY

- Taxonomy: hierarchy of subclasses
- Subcategory relations organize categories into a **taxonomy** or **taxonomic hierarchy.**
  - Other names are type hierarchy or class hierarchy .
- We state that a category is a subcategory of another category by using the notation for subsets

  Basketball $\subset$ Ball

- We will also use the notation
- **ako(basketball,ball).**

INHERITANCE

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

TYPES OF FACTS

- An object is a member of a category.
  - BB9 ∈ Basketballs
- A category is a subclass of another category.
  - Basketballs ⊂ Balls
- All members of a category have some properties.
  - (x ∈ Basketballs) ⟹Spherical (x)
- Members of a category can be recognized by some pr operties.
  - Orange(x)∧ Round (x) ∧ Diameter(x)=9.5'' ∧ x ∈ Balls ⟹ x ∈ Basketballs
- A category as a whole has some properties.
  - Dogs ∈ DomesticatedSpecies (category of categories)

CATEGORY DECOMPOSITIONS

- Because categories are sets, we handle them as such.
  - e.g., two categories are disjoint if they have no member in common
- A disjoint exhaustive decomposition is called a partition
  - Ex> We can say that both Male and Female is a subclass of Animal, but we have not said that a male cannot be a female.
    - Disjoint({Males, Females})
  - If we know that all animals are either male or female, (they exhaust the possibilities)
    - ExhaustiveDecomposition({Male,Female},Animals)
  - A disjoint exhaustive decomposition is known as a partition
    - Partition({Males, Females},Animals)
- Categories can also be *defined* by providing necessary and suff icient conditions for membership.
  - For example, a bachelor is an unmarried adult male:
  - $x \in Bachelors \Leftrightarrow Unmarried(x) \land x \in Adults \land x \in Males$.

COMPOSITE OBJECTS

- Composite object: any object that has parts
- One object can be a part of another **object.**
  - Example, declaring direct parts
    - **partof(bucharest,romania).**
    - **partof(romania,eastern_europe).**
    - **partof(europe,earth).**
- We can make a transitive extension partof
  - **partof(Y,Z) and partof(X,Y) => partof(X,Z).**
- and reflexive (depending on definition)
  - **partof(X,X).**
- Therefore we can conclude that
  - **partof(bucharest,earth)**

COMPOSITE OBJECTS

- Categories of composite objects often characterized by their structure, i.e., what the parts are and how they relate.

$$
\begin{aligned}
\text{e.g., } & \forall a\ Biped(a) \Rightarrow \\
& \exists\ ll, lr, b \\
& Leg(ll) \wedge Leg(lr) \wedge Body(b) \wedge \\
& PartOf(ll, a) \wedge PartOf(lr, a) \wedge PartOf(b, a) \wedge \\
& Attached(ll, b) \wedge Attached(lr, b) \wedge \\
& ll \neq lr \wedge \\
& \forall x\ Leg(x) \wedge PartOf(x, a) \Rightarrow (x = ll \vee x = lr)
\end{aligned}
$$

- Such description can be used to describe any objects, including events.

* Biped : a two-footed animal.

PHYSICAL COMPOSITIONS: BUNCHOF

- It is also useful to define composite objects with definite parts but no particular structure.
  - Ex> *"The apples in the bag weigh two pounds"*
- It is advised that we don't regard these apples as the set of (all) apples, but instead define them as a **bunch of** apples.
  - EX>If the apples are Apple1,Apple2 and Apple3, then

    BunchOf({Apple1,Apple2,Apple3})
- Denotes the composite object with three apples as **parts,** not elements.

MEASUREMENTS : QUANTITATIVE

- Objects have height, mass, cost and so on. The values we assign for these properties are called **measures.**
- Can be represented using **units functions**
  - E.g.> Length(L1) = Inches(1.5) = Centimeters(3.81)
- Measures can be used to describe objects
  - e.g.> Mass(Tomato12) = Kilograms(0.16)

$$\forall b,\ b \in DollarBills \Rightarrow CashValue(b) = \$(1.00)$$

- Caution: be careful to distinguish between measures and objects
  - e.g.>

MEASUREMENTS

- **Abstract** concepts like ”autonomy”, ”quality” and **Mental** concepts are beliefs, thoughts, feelings etc. are difficult to represent without seeking artificial measurements. (e.g. IQ).
- However, the most important aspect of measueres is no the particular numerical valuesm but the fact that measures can be **ordered**.

EVENTS

- **Events** are chunks of spatio-temporal universe
  - e.g., consider the event WorldWarII

  it has parts or sub-events:

  SubEvent(BattleOfBritain, WorldWarII)

  it can be a sub-event:

  SubEvent(WorldWarII, TwentiethCentury)
- **Intervals** are events that include as sub-events all events occurring in a given time period (thus they are temporal sections of the entire spatial universe).

SITUATION CALCULUS VS EVENT CALCULUS:

- situation calculus:
  - fact true in particular situation
  - can't talk about what happens during the action and
  - can't describe two actions happening at the same time.
- event calculus:
  - based on points of time
  - event occurs during particular interval,
  - focus on how to deal with change based on representing points of time rather than on situation

T(f, t)	Fluent *f* is true at time *t*
Happens(e, i)	Event *e* happens over interval *i*
Initiates(e, f, t)	Event *e* causes fluent *f* to start at *t*
Terminates(e, f, t)	Event *e* causes *f* to cease at *t*
Clipped(f, t)	Fluent *f* ceases to be true in int. *i*
Restored(f, i)	Fluent *f* becomes true in interval *i*

EVENTS (CONT.)

- Define *T* by saying that a
  - *fluent holds at a point in time* if the fluent was initiated by an event at some time in the past and was not made false (clipped) by an intervening event.
  - *a fluent holds over an interval* if it holds on every point within the interval
  - *fluent does not hold* if it was terminated by an event and not made true (restored) by another event.
- Places: spatial sections of the spatio-temporal universe that extend through time
- Use In(x) to denote sub-event relation between places: e.g. In(N
  $\forall x,l\ Location(x) = l \Leftrightarrow At(x, l) \wedge \forall ll\ At(x, ll) \Rightarrow In(l, ll)$
- Location function: maps an object to the smallest place that contains it:

PROCESSES

- Event we have seen so fare are discrete events.
  - They have a definite structure.
- Ones that are not discrete are **process categories (liquid events)**.
  - Any process e that happens over an interval also happens over any subinterval.
  - ex> *Flyings :* If we take a small interval of Shankar's flight, s ay, the third 20-minute segment (while he waits anxiously for a bag of peanuts), that event is still a member of *Flyings*.

TIMES INTERVALS AND ACTIONS

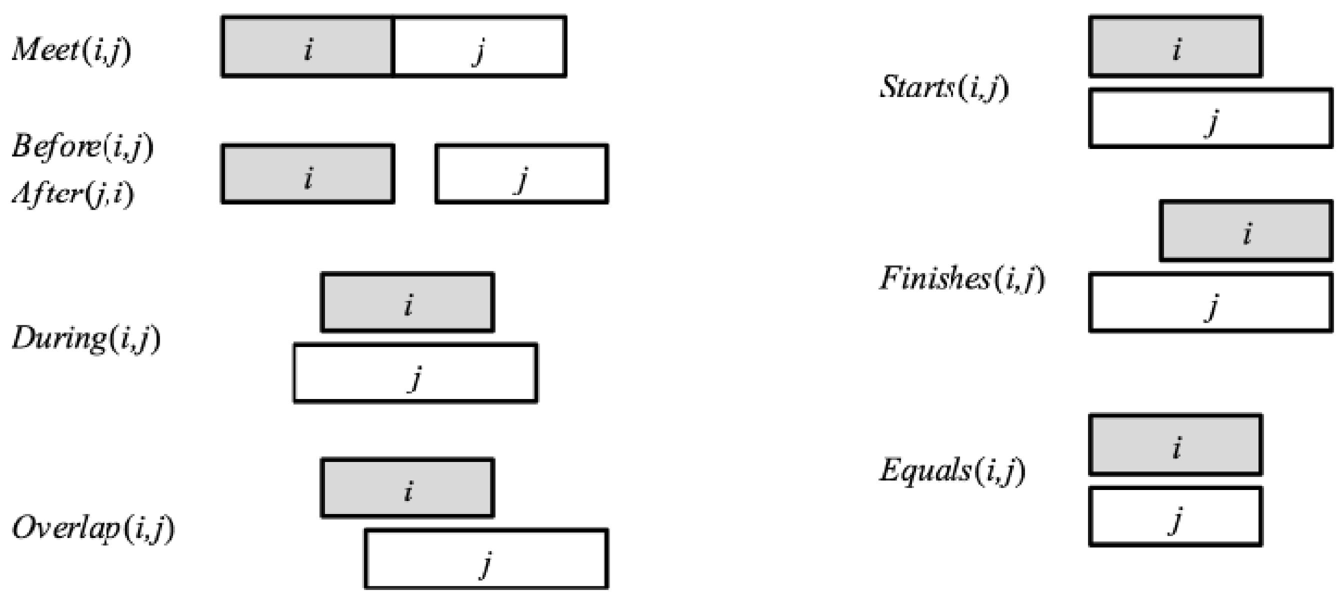
- Time intervals can be partitioned between moments (=zero duration) and extended intervals:
- Absolute times can then be derived from defining a time scale and associating points on that scale with events.
  - e.g., seconds since midnight GMT on Jan 1, 1900
- The functions **Start** and **End** then pick the earliest and latest moments in an interval.
- The function **Duration** gives the difference between end and start times.

∀i Interval(i) ⇒ Duration(i) = (Time(End(i) − Time(Start(i)))
Time(Start(AD1900)) = Seconds(0)
Time(Start(AD1991)) = Seconds(2871694800)
Time(End(AD1991)) = Seconds(2903230800)
Duration(AD1991) = Seconds(31536000)

PREDICATES IN TIME INTERVALS

$$
\begin{array}{lll}
Meet(i,j) & \Leftrightarrow & End(i) = Begin(j) \\
Before(i,j) & \Leftrightarrow & End(i) < Begin(j) \\
After(j,i) & \Leftrightarrow & Before(i,j) \\
During(i,j) & \Leftrightarrow & Begin(j) < Begin(i) < End(i) < End(j) \\
Overlap(i,j) & \Leftrightarrow & Begin(i) < Begin(j) < End(i) < End(j) \\
Begins(i,j) & \Leftrightarrow & Begin(i) = Begin(j) \\
Finishes(i,j) & \Leftrightarrow & End(i) = End(j) \\
Equals(i,j) & \Leftrightarrow & Begin(i) = Begin(j) \wedge End(i) = End(j)
\end{array}
$$

set of interval relations, as proposed by Allen (1983)



FLUENTS AND OBJECTS REVISITED

- It is legitimate to describe physical objects as generalized events,
  - In the sense that a physical object is a chunk of space-time
  - e.x. *USA* can be thought of as an event that began in 1776 as a union of 13 states and is still in progress today as a union of 50.
- We can then use temporal and spatial sub-events to capture changing properties of the objects

  e.g., Poland: event

  e.g., 19thCenturyPoland: temporal sub-event

  e.g., CentralPoland: spatial sub-event
- We call fluents objects that can change across situations
  - Describe the changing properties of *USA* using state fluents, such as *Population(USA)*.

SUBSTANCES AND OBJECTS

- Some objects cannot be divided into distinct parts –
  - e.g., butter: one butter? no, some butter!
  - butter substance (and similarly for temporal substances)
  - (simple rule for deciding what is a substance: if you cut it in half, you should get the same).
- How can we represent substances?
  - Start with a category

    e.g., $\forall x,y \quad x \in Butter \wedge PartOf(y, x) \Rightarrow y \in Butter$
  - Then we can state properties

    e.g., $\forall x \; Butter(x) \Rightarrow MeltingPoint(x, Centigrade(30))$

- Need to distinguish between substance and discrete objects
- Substance (“stuff”)
  - Mass nouns - not countable
  - Intrinsic properties
  - Part of a substance is (still) the same substance
- Discrete objects (“things”)
  - Count nouns - countable
  - Extrinsic properties
  - Parts are (generally) not of same category

MENTAL EVENTS AND MENTAL OBJECTS

- Need to represent *beliefs* in self and other agents, e.g. for controlling reasoning, or for planning actions that involve others
- How are beliefs represented?
  - Beliefs are reified as *mental objects*
  - Mental objects are represented as *strings* in a language
  - Inference rules for this language can be defined
- Rules for reasoning about logical agents' use their beliefs

$$\forall a,p,q \ LogicalAgent(a) \wedge Believes(a,p) \wedge Believes(a,"p \Rightarrow q") \Rightarrow Believes(a,q)$$

$$\forall a,p \quad LogicalAgent(a) \wedge Believes(a,p) \Rightarrow Believes(a,"Believes(Name(a),p)")$$

REASONING SYSTEMS FOR CATEGORIES

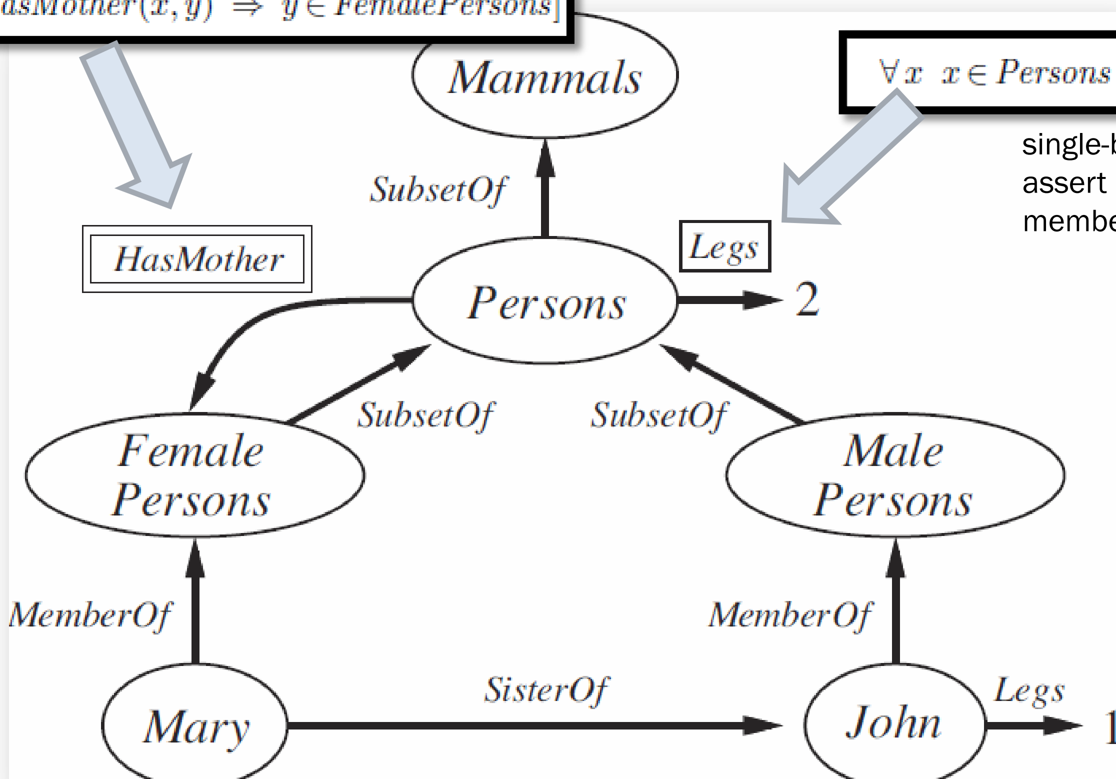
- Semantic networks graphical representation for visualizing
  - a knowledge base and
  - efficient algorithms for inferring properties of an object on the basis of its category membership
- Description logics is a formal language for
  - constructing and combining category definitions and
  - efficient algorithms for deciding subset and superset relationships between categories.

SEMANTIC NETWORKS

- There are many variants of semantic networks, but all are capable of representing individual <u>objects, categories of objects, and relations among objects</u>
- Natural representation of inheritance and default values

$\forall x \;\; x \in Persons \Rightarrow [\forall y \;\; HasMother(x, y) \Rightarrow y \in FemalePersons]$

$\forall x \;\; x \in Persons \Rightarrow Legs(x, 2)$



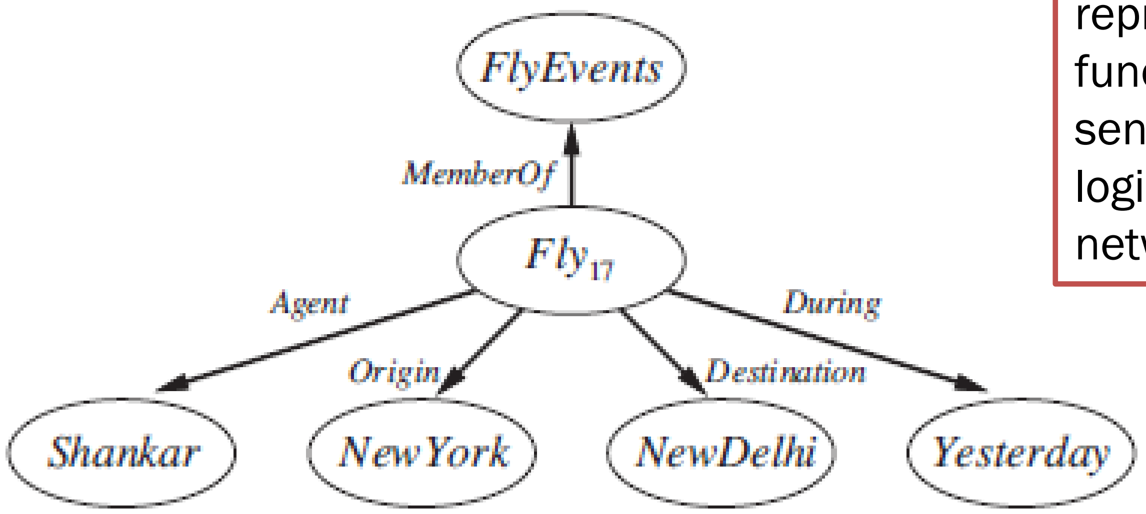
single-boxed link used to assert properties of every member of a category

NOTE: links between bubbles represent only *binary* relations

SEMANTIC NETWORKS CONT.

- We *can* obtain the effect of n-ary assertions by reifying the proposition itself as an event belonging to an appropriate event category.

Reification of propositions makes it possible to represent every ground, function-free atomic sentence of first-order logic in the semantic network notation



A fragment of a semantic network showing the representation of the logical assertion *Fly(Shankar , NewYork, NewDelhi , Yesterday)*.

DESCRIPTION LOGICS

- Derived from semantic networks, but more formal
- For inference supports
  - **Subsumption:** checking if one category is a subset of another by comparing their definition
  - **Classification:** checking whether an object belongs to a category
  - **Consistency:** whether the membership criteria are logically satisfiable.
- But still not applicable to large problems

CLASSIC LANGUAGE

- The CLASSIC language (Borgida *et al.*, 1989) is a typic al description logic

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

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

Example

$$Bachelor = And(Unmarried, Adult, Male)\ .$$

The equivalent in first-order logic would be

$$Bachelor(x) \Leftrightarrow Unmarried(x) \wedge Adult(x) \wedge Male(x)\ .$$

EXAMPLE: INTERNET SHOPPING WORLD

- An agent that understands and acts in an internet shopping environment
- The task is to shop for a product on the Web, given the user's product description
- The product description may be **precise**, in which case the agent should find the **best price**
- In other cases the description is only **partial**, and the agent has to **compare products**
- The shopping agent depends on having **product knowledge, incl. category hierarchies**

PEAS SPECIFICATION OF SHOPPING AGENT

- Performance goal
  - Recommend product(s) to match user's description
- Environment
  - All of the Web
- Actions
  - Following links
  - Retrieve page contents
- Sensors
  - Web pages: HTML, XML

OUTLINE OF AGENT BEHAVIOR

- Start at home page of known web store(s)
  - Must have knowledge of relevant web addresses
- Spread out from home page, **following links** to relevant pages containing product offers
  - Must be able to identify page relevance, using product category ontologies, as well parse page contents to detect product offers
- Having located one or more product offers, agent must **compare offers** and recommend product
  - Comparison range from simple price ranking to complex tradeoffs in several dimensions

- Find relevant product offers
  - RelevantOffer(page,url,query) $\Leftrightarrow$ Relevant(page, url, query) $\land$ Offer(page)
  - Write axioms to define Offer(x)
  - Find relevant pages: Relevant(x,y,z) ?
    - Start from an initial set of stores.
    - What is a relevant category?
    - What are relevant connected pages?
  - Require rich category vocabulary.
    - Synonymy and ambiguity
  - How to retrieve pages: *GetPage(url)*?
    - Procedural attachment
- Compare offers (information extraction).

EXAMPLE OF RICH CATEGORY VOCABULARY

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

(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.

EXAMPLE OF FOLLOWING LINKS

- The agent will have knowledge of a number of stores

$$Amazon \in OnlineStores \wedge Homepage(Amazon, \text{"amazon.com"}) .$$
$$Ebay \in OnlineStores \wedge Homepage(Ebay, \text{"ebay.com"}) .$$
$$ExampleStore \in OnlineStores \wedge Homepage(ExampleStore, \text{"example.com"})$$

- A page is relevant to the query if it can be reached by a chain of zero or more relevant category links from a store's home page, and then from one more link to the product offer.

$$\begin{aligned} &Relevant(page, query) \Leftrightarrow \\ &\quad \exists\, store, home \;\; store \in OnlineStores \wedge Homepage(store, home) \\ &\quad \wedge \exists\, url, url_2 \;\; RelevantChain(home, url_2, query) \wedge Link(url_2, url) \\ &\qquad \wedge page = Contents(url) . \end{aligned}$$

$$\begin{aligned} &RelevantChain(start, end, query) \Leftrightarrow (start = end) \\ &\quad \vee (\exists\, u, text \;\; LinkText(start, u, text) \wedge RelevantCategoryName(query, text) \\ &\qquad \wedge RelevantChain(u, end, query)) . \end{aligned}$$

$$\begin{aligned} &RelevantCategoryName(query, text) \Leftrightarrow \\ &\quad \exists\, c_1, c_2 \;\; Name(query, c_1) \wedge Name(text, c_2) \wedge (c_1 \subseteq c_2 \vee c_2 \subseteq c_1) . \end{aligned}$$

EXAMPLE OF COMPARING OFFERS

- Wrapper to extract information

  Given a page on the example.com site with the text

```
IBM ThinkBook 970.  Our price:  $399.00
```

$$\exists\, c, \mathit{offer} \;\; c \in \mathit{LaptopComputers} \land \mathit{offer} \in \mathit{ProductOffers} \land \\ \mathit{Manufacturer}(c, \mathit{IBM}) \land \mathit{Model}(c, \mathit{ThinkBook970}) \land \\ \mathit{ScreenSize}(c, \mathit{Inches}(14)) \land \mathit{ScreenType}(c, \mathit{ColorLCD}) \land \\ \mathit{MemorySize}(c, \mathit{Gigabytes}(2)) \land \mathit{CPUSpeed}(c, \mathit{GHz}(1.2)) \land \\ \mathit{OfferedProduct}(\mathit{offer}, c) \land \mathit{Store}(\mathit{offer}, \mathit{GenStore}) \land \\ \mathit{URL}(\mathit{offer}, \text{“}\mathtt{example.com/computers/34356.html}\text{”}) \land \\ \mathit{Price}(\mathit{offer}, \$(399)) \land \mathit{Date}(\mathit{offer}, \mathit{Today})\,.$$

- Compare the offers that have been extracted

$$A : 1.4 \text{ GHz CPU, 2GB RAM, 250 GB disk, } \$299\,. \\ B : 1.2 \text{ GHz CPU, 4GB RAM, 350 GB disk, } \$500\,. \\ C : 1.2 \text{ GHz CPU, 2GB RAM, 250 GB disk, } \$399\,.$$