GENERAL ONTOLOGY

Dr. Sunil Surve

Fr. Conceicao Rodrigues College of Engineering

General Ontology

- Incorporates decisions about how to represent a broad selection of objects and relations
- It is encoded within first-order logic, but makes many ontological commitments that first-order logic does not make
- Characteristics of general-purpose ontologies
  - A general-purpose ontology should be applicable in more or less any special-purpose domain. For example, a general ontology cannot use situation calculus, which finesses the issues of duration and simultaneity
  - In any sufficiently demanding domain, different areas of knowledge must be *unified* because reasoning and problem solving may involve several areas simultaneously. A robot circuitrepair system, for instance, needs to reason about circuits in terms of electrical connectivity and physical layout, and about time both for circuit timing analysis and estimating labor costs.

General Ontology

- **Categories:**
  - Rather than being an entirely random collection of objects, the world exhibits a good deal of regularity
  - Define **categories** that include as members all objects having certain properties
  - How categories can be objects in their own right, and how they are linked into a unified **taxonomic hierarchy**
- **Measures:**
  - Many useful properties such as mass, age, and price relate objects to quantities of particular types, which we call measures

General Ontology

- **Composite objects:**
  - It is very common for objects to belong to categories by virtue of their constituent structure. For example, cars have wheels, an engine, and so on, arranged in particular ways
- **Time, Space, and Change:**
  - In order to allow for actions and events that have different durations and can occur simultaneously, we enlarge our ontology of time
- **Events and Processes:**
  - Events such as the purchase of a tomato will also become individuals in our ontology. Like tomatoes, they are usually grouped into categories. Individual events take place at particular times and places

General Ontology

- **Physical Objects:**
  - We are already familiar with the representation of ordinary objects such as AND-gates and wumpuses
- **Substances:**
  - Whereas objects such as tomatoes are relatively easy to pin down, substances such as tomato juice are a little slippery
- **Mental Objects and Beliefs:**
  - An agent will often need to reason about its own beliefs, for example, when trying to decide why it thought that anchovies were on sale
  - Sentences are explicitly represented, and are believed by agents

Representing Categories

- Although interaction with the world takes place at the level of individual objects, *much of reasoning* takes place at the level of categories
- Categories also serve to make predictions about objects once they are classified
- One 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
- Categories are represented by unary predicates. For example, Tomato

Representing Categories

- **Reification** is the process of turning a predicateor function into an object in the language
- *Tomatoes* as a constant symbol referring to the object that is the *set of all tomatoes.*
- We use *x* G *Tomatoes* to say that *x* is a tomato.
- Reified categories allow us to make assertions about the category itself, rather than about members of the category.
- For example, we can say *Population(Humans)=5,*000,000,000
- Categories serve to organize and simplify the knowledge base through **inheritance**
- Subclass relations organize categories into a taxonomy or taxonomic hierarchy

Representing 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. For example:
Tomatou G *Tomatoes*
A category is a subclass of another category. For example:
Tomatoes $\subset$ *Fruit*

- All members of a category have some properties. For example:
$\forall$*x x* G *Tomatoes* $\Rightarrow$ *Red(x)* Λ *Round(x)*

- Members of a category can be recognized by some properties. For example: $\forall$*X Red(Interior(x))* Λ *Green(Exterior(x))*Λ *x* G *Melons*
$\Rightarrow$ *x* $\in$ *Watermelons*

- A category as a whole has some properties. For example:
Tomatoes G *DomesticatedSpecies*

Representing Categories

- Two or more categories are **disjoint** if they have no members in common.
- A disjoint **exhaustive decomposition** is known as a **partition**
- Disjoint({Animals, Vegetables})
- ExhaustiveDecomposition({Americans, Canadians, Mexicans}, NorthAmericans)
- Partition({ Males, Females], Animals)
- $\forall$s *Disjoints (s)* $\Leftrightarrow$ ($\forall$ c1, c2 *c1* $\in$ *s* Λ c2 $\in$ s Λ *c1* $\neq$c2 $\Rightarrow$ Intersection(c1, c2) = EmptySet()
$\forall$*s, c ExhaustiveDecomposition(s, c)* $\Leftrightarrow$ ($\forall$i *i*$\in$*c* $\Leftrightarrow$ $\exists$c2 c2$\in$ s Λ i $\in$ c2)
$\forall$*s, c Partition (s, c)* $\Leftrightarrow$ *Disjoint(s)* Λ *ExhaustiveDecomposition(s, c)*
- *Categories can also be defined by providing necessary and sufficient conditions for membership*
$\forall$*x Bachelor(x)* $\Leftrightarrow$ *Male(x)* Λ *Adult(x)* Λ *Unmarried(x)*

Measures

- *Length(L1) = Inches(* 1.5) = *Centimeters(3.8*1)
- Conversion between units is done with sentences such as
$\forall l$ *Centimeters(2.54* x *l)=Inches(l)*
$\forall t$ *Centigrade(t)=Fahrenheit(32+* 1.8 x *t)*
- Measures can be used to describe objects as follows:
Mass(Tomato12) = Kilograms(0.16)
Price(Tomato12) = \$(0.32)
$\forall d\ d \in Days \Rightarrow Duration(d) = Hours(24)$
- It is very important to be able to distinguish between monetary amounts and monetary instruments:
$\forall b\ b \in DollarBills \Rightarrow CashValue(b) = \$(l.OO)$

Measures

- The most important aspect of measures is not the particular numerical values, but the fact that measures can be *ordered*
- For example, we might well believe that Norvig's exercises are tougher than Russell's, and that one scores less on tougher exercises:
$\forall e1, e2\ e1 \in Exercises \wedge e2 \in Exercises \wedge Wrote(Norvig, e1) \wedge Wrote(Russell, e2) \Rightarrow Difficulty(e1) > Difficulty(e2)$

$\forall e1, e2\ e1 \in Exercises \wedge e2 \in Exercises \wedge Difficulty(e1) > Difficulty(e2) \Rightarrow ExpectedScore(e1) < ExpectedScore(e2)$

Composite objects

- One object can be part of another
- Objects can therefore be grouped into *PartOf* hierarchies, reminiscent of the *Subset* hierarchy:
PartOf (Bucharest, Romania)
PartOf (Romania, EasternEurope)
PartOf (EasternEurope, Europe)
- Any object that has parts is called a **composite object.**
- Categories of composite objects are often characterized by the **structure** of those objects
$\forall$a Biped(a) $\Rightarrow$ $\exists$l1,l2,b Leg(l1) Λ Leg(l2) Λ Body(b) Λ PartOf(l1,a) Λ PartOf(l2, a) Λ PartOf(b, a) Λ Attached(l1,b) A Attached(l2, b) Λ L1 $\neq$ l2 Λ l3 Leg(l3) Λ PartOf(l3,a $\Rightarrow$ (l1 = l3 V l3 = l2)

Composite objects

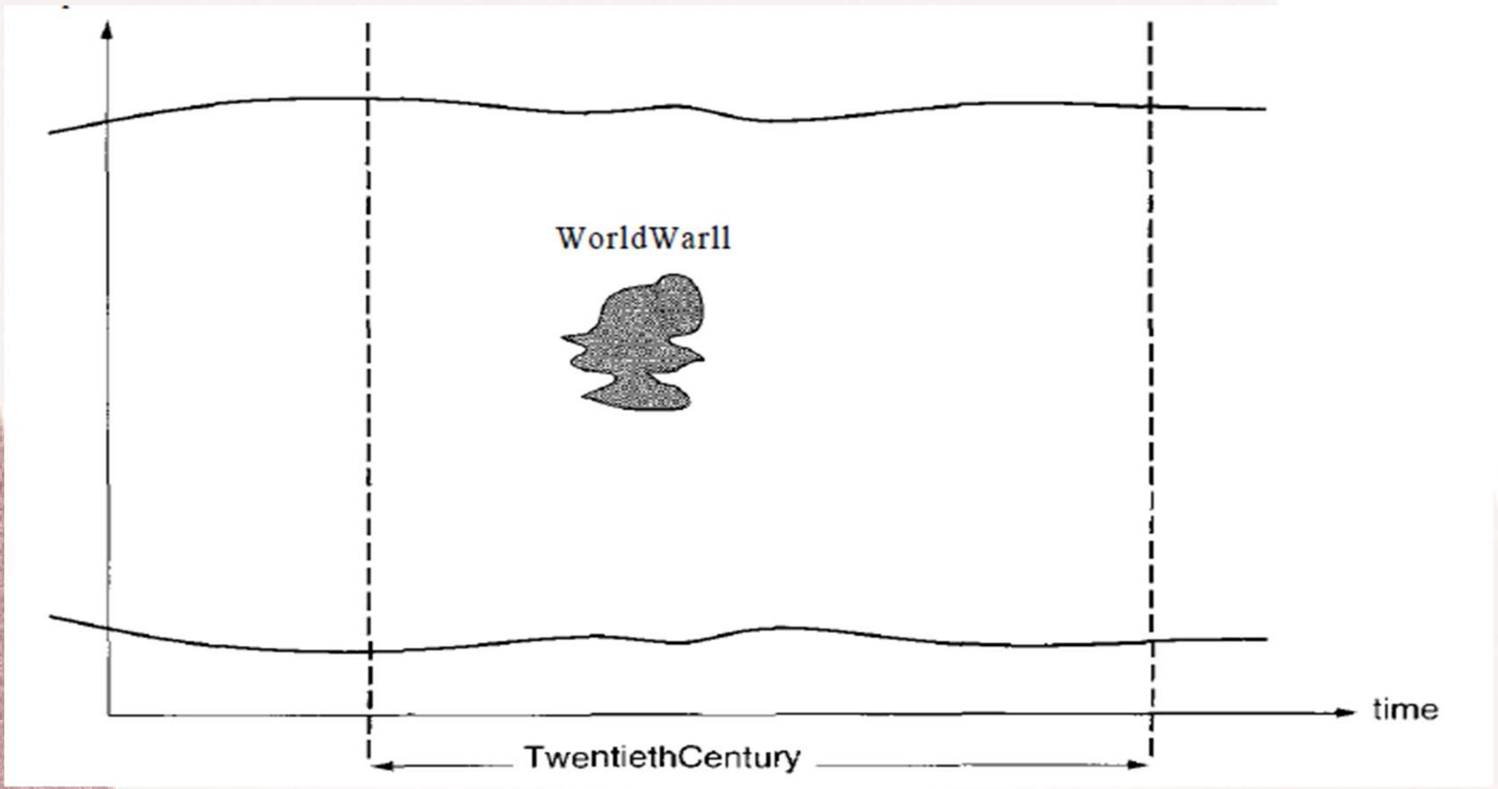
- A generic event description like games is often called a **schema** or **script**
- An object is composed of the parts in its *PartPartition,* and can be viewed as deriving some properties from those parts
- For example, the mass of a composite object is the sum of the masses of the parts
- Notice that this is not the case with categories: categories have no mass, even though their elements might
- It is also useful to define composite objects with definite parts but no particular structure
- **Bunch** out of the apples: if the apples are *Apple1* , *Apple2,* and *Apple3,* then
BunchOf({Apple1 , Apple2, Apple3})

Representing change with events

- Situations are instantaneous points in time, which are not very useful for describing the gradual growth of a kitten into a cat, the flow of electrons along a wire, or any other process where change occurs continuously over time.
- Situation calculus works best when only one action happens at a time
- When there are multiple agents in the world, or when the world can change spontaneously, situation calculus fails
- Event calculus is rather like a continuous version of the situation-calculus
- The "spatial" dimension ranges over all of the objects in an instantaneous "snapshot" or "cross-section" of the universe
- The temporal dimension ranges over time

Representing change with events

- An **event** is, informally, just a "chunk" of this universe with both temporal and spatial extent
- SubEvent(BattleOfBritain, WorldWarll) SubEvent(WorldWarll, TwentiethCentury)



Representing change with events

- An interval is an event that includes as subevents all events occurring in a given time period.
- Intervals are therefore entire temporal sections of the universe
- In event calculus, a given event occurs during a particular interval
- $\exists$ *w w* G *Wars* Λ *SubEvent(w, AD* 1967) Λ *PartOf(Location(w), MiddleEast)*
  $\exists$*j j E Journeys* Λ *Origin(NewYork,j)* Λ *Destination(NewDelhi,j)* Λ *Traveller(Shankar,j)* Λ *SubEvent(j, Yesterday)*
  $\forall$*e, x, o,d e* $\in$ *Go(x, o, d)* $\Rightarrow$ $\exists$*e e Journeys* Λ *Traveller(x, e)* Λ *Origin(o, e)* Λ *Destination(d, e)*
- *The notation E(c, i) to say that an event of category c is a subevent of the event (or interval):* $\forall$ *c, i E(c, i)* $\Leftrightarrow$ $\exists$*e e* $\in$ *c* Λ *SubEvent(e, i)*
  E(Go(Shankar, NewYork, NewDelhi), Yesterday)

Places

- **Places,** like intervals, are special kinds of space-time chunks
- A place can be thought of as a constant piece of space, extended through time
In(NewYork, USA)
- Places come in different varieties; for example, *NewYork* is an *Area,* whereas the *SolarSystem* is a *Volume*
- $\forall x,l\ Location(x) = l \Leftrightarrow At(x, l) \lor \forall l2\ At(x,l2) \Rightarrow In(l,l2)$
- This last sentence is an example of a standard logical construction called **minimization**

Processes

- **Event** has a beginning, middle, and end. If interrupted halfway, the event would be different
- Categories of events with this property are called **process** categories or **liquid event** categories
- Any subinterval of a process is also a member of the same process category
E(Flying(Shankar), Yesterday)
- We often want to say that some process was going on *throughout* some interval, rather than just in some subinterval of it. T(Working(Stuart), TodayLunchH our)
- *T(c. i)* means that some event of type *c* occurred over exactly the interval *i*—that is, the event begins and ends at the same time as the interval

Processes

- As well as describing processes of continuous change, liquid events can describe processes of continuous non-change. These are often called **states**
- In(Mary, Supermarket])
- T(In(Mary, Supermarket]), ThisAfternoon)
- T(Closed(Superinarket\), BunchOf (Sundays))

Times, intervals, and actions

- Time intervals are partitioned into moments and extended intervals
- Partition({Moments, Extendedfntervals}, Intervals)
  $\forall i\ i \in$ Intervals $\Rightarrow$ (i G Moments $\Leftrightarrow$ Duration(i) =G)
- The functions *Start* 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
- The function *Duration* gives the difference between the end time and the start time
- $\forall$i Interval(i) $\Rightarrow$ Duration(i)=(Time(End(i)) -Time(Start(i)))
  Time(Start(ADl900)) =Seconds(0)
  Time(Start(AD 1991)) =Seconds(2811694800)
  Time(End(AD 1991)) = Secon<fr(2903230800)
  Duration(AD 1991) =Seconds(31536000)

Times, intervals, and actions

- Two intervals *Meet* if the end time of the first equals the start time of the second
- $\forall$i,j Meet(i,j) $\Leftrightarrow$ Time(End(i)) = Time(Start(j))
- $\forall$ i,j Before(i,j) $\Leftrightarrow$ Time(End(i)) < Time(Start(j))
- $\forall$i,j After(j,i) $\Leftrightarrow$ Before(i, j)
- $\forall$i,j During(i, j) $\Leftrightarrow$ Time(Start(j)) < Time(Start(i)) Λ Time(End(i)) < Time(End(j)
- $\forall$i,j Overlap(i, j) $\Leftrightarrow$ $\exists$k During(k, i) Λ During(k, j)

Times, intervals, and actions

- Temporal relations among intervals are used principally in describing actions
- If two people are engaged, then in some future interval, they will either marry or break the engagement.
  $\forall$x,y,i0 T(Engaged(x,y),i0) $\Rightarrow$ $\exists$i1 (Meet(i0, i1) V After(i1, i0)) Λ T(Marry(x,y) V BreakEngagement(x,y), i1)
- When two people marry, they are spouses for some interval starting at the end of the marrying event
  $\forall$x,y,i0 T(Marry(x,y),i0) $\Rightarrow$ $\exists$i1 T(Spouse(x, y), i1) Λ Meet(i0, i1)
- The result of going from one place to another is to be at that other place
  $\forall$x,a.b,i0, $\exists$i1 T(Go(x,a,b),i0) $\Rightarrow$ T(In(x,b),i1) Λ Meet(i0,i1)

Substances and objects

- **Stuff**: A significant portion of reality that seems to defy any obvious **individuation**—division into distinct objects
- Example of stuff and things: Butter, Apple
- In fact, some have called liquid event types **temporal substances,** whereas things like butter are **spatial substances**
- Linguists distinguish between **count nouns,** such as aardvarks, holes, and theorems, and **mass nouns,** such as butter, water, and energy
- Any part of a butter-object is also a butter-object:
  $\forall$ x, y x G Butter A PartOf(y, x) $\Rightarrow$ y $\in$ Butter
- Butter melts at around 30 degrees centigrade:
  $\forall$x Butter(x) $\Rightarrow$ MeltingPoint(x,Centigrade(30))

Substances and objects

- Butter is yellow, less dense than water, soft at room temperature, has a high fat content, and so on
- On the other hand, butter has no particular size, shape, or weight
- PoundOfButter
- **Intrinsic properties –** density, boiling point, flavor, color, ownership
- **Extrinsic properties –** weight, length, shape, function
- A class 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
- *Stuff* is the most general substance category, specifying no intrinsic properties
- *Thing* is the most general discrete object category, specifying no extrinsic properties

Substances and objects

- An object belongs to both mass and count classes. For example, *LakeMichigan* is an element of both *Water* and *Lakes*
- Water in Lake Michigan changes over time, simply by viewing the lake as an event whose constituent objects change over time

Mental events and mental objects

- The agents have beliefs and can deduce new beliefs
- For single-agent domains, knowledge about one's own knowledge and reasoning processes is useful for controlling inference
- In 'multiagent domains, it becomes important for an agent to reason about the mental processes of the other agents
- Shopper in a supermarket has the goal of buying some salt: Plan, execute, fails, inquire with employee, etc.
- 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
- An abstract model that says that if a logical agent believes $P \vee Q$ and it learns $\neg$P, then it should come to believe Q

Mental events and mental objects

- If we have a relation *Believes(Agent,x),* what kind of thing is *x*?
- Flies(Superman)
- Believes(Agent, Flies(Superman))
- **Propositional attitudes:** *Knows* and *Wants* to express other relationships between agents and propositions
- (Superman = Clark) |= (Believes(Lc)is, Flies(Superman)) ⇔ Believes(Lois, Flies(Clark)))
- The property of being able to freely substitute a term for an equal term is called referential transparency
- In first-order logic, every relation is referentially transparent
- Referentially **opaque**—that is, one cannot substitute an equal term for the second argument without changing the meaning

Mental events and mental objects

- Den("Clark") - ManOfSteel Λ Den("Superman") = ManOfSteel Name(ManOfSteel) -"K1,"
- $\forall$ a,p, q LogicalAgent(a) A Believes(a,p) Λ Believes(a, "p $\Rightarrow$ q") $\Rightarrow$ Believes(a, q) X
- $\forall$ a,p, q LogicalAgent(a) A Believes(a,p) $\forall$ Concat(p, $\Rightarrow$, q) $\Rightarrow$ Believes(a, q)
- Besides the normal inference rules, we need some rules that are specific to belief
- $\forall$a, p LogicalAgent(a) $\forall$ Believes(a,p) $\Rightarrow$ Believes(a,"Believes(Name(a),p)")

Mental events and mental objects

- **Logical omniscience:** an agent can, according to our axioms, deduce any valid conclusion instantaneously
- Define axioms for other propositional attitudes
- $\forall$a,p Knows(a,p) $\Leftrightarrow$ Believes(a,p) Λ T(Den(p)) Λ T(Den(KB(a)) $\Rightarrow$ Den(p))
- $\forall$a,p KnowsWhether(a,p) $\Leftrightarrow$ Knows(a,p) V Knows(a,"$\neg$p")
- $\forall$a,b KnowsWhat(a,"PhoneNumber(b)
  $\Leftrightarrow$ $\exists$x Knows(a, "x = PhoneNumber(b)") Λ DigitString(x)
- KnowsWhat(Agent, Capital(NewYork), ProperName)
  KnowsWhat(Agent, PhoneNumber(Bob), DigitString)

Mental events and mental objects

- To recognize that propositional attitudes change over time
- T(process, interval)
- Believe(agent, string, interval)
- Lois believed yesterday that Superman can fly:
Believes(Lois, Flies(Superman), Yesterday)
- It will be true tomorrow that Lois knew that Superman could fly yesterday:
T(Believes(Lois, Flies(Superman), Yesterday), Tomorrow)
- It is true now that Jimmy knows today that Lois believes that Superman could fly yesterday:
T(Knows(Jimmy, Believes(Lois, Flies(Superman), Yesterday), Today), Now)

THE GROCERY SHOPPING WORLD

- **Percepts**:
  - The agent receives three percepts at each time step: feel, sound, and vision
  - The feel percept is just a bump or no bump, as in the vacuum world. The agent perceives a bump only when on the previous time step it executed a *Forward* action and there is not enough room in the location it tried to move to
  - The sound percept is a list of spoken words. The agent perceives words spoken by agents within two squares of it
  - If the agent's camera is zoomed in, it perceives detailed visual images of each object in the square it is zoomed at
  - If the agent's camera is not zoomed in, it perceives coarse visual images of each object in the three squares directly and diagonally ahead
  - A visual percept consists of a relative location, approximate size, color, shape, and possibly some other features

THE GROCERY SHOPPING WORLD

- **Actions**:
  - An agent can speak a string of words
  - An agent can go one square forward
  - An agent can turn 90° to the right or left
  - An agent can zoom its camera in at its current square, or at any of the three squares directly or diagonally ahead
  - An agent can also zoom its camera out
  - An agent can grab an object that is within one square of it. To do so, it needs to specify the relative coordinates of the object, and it needs to be empty-handed
  - An agent can release an object that it has grabbed. To do so, it needs to specify the relative coordinates of the point where it wants to release the object

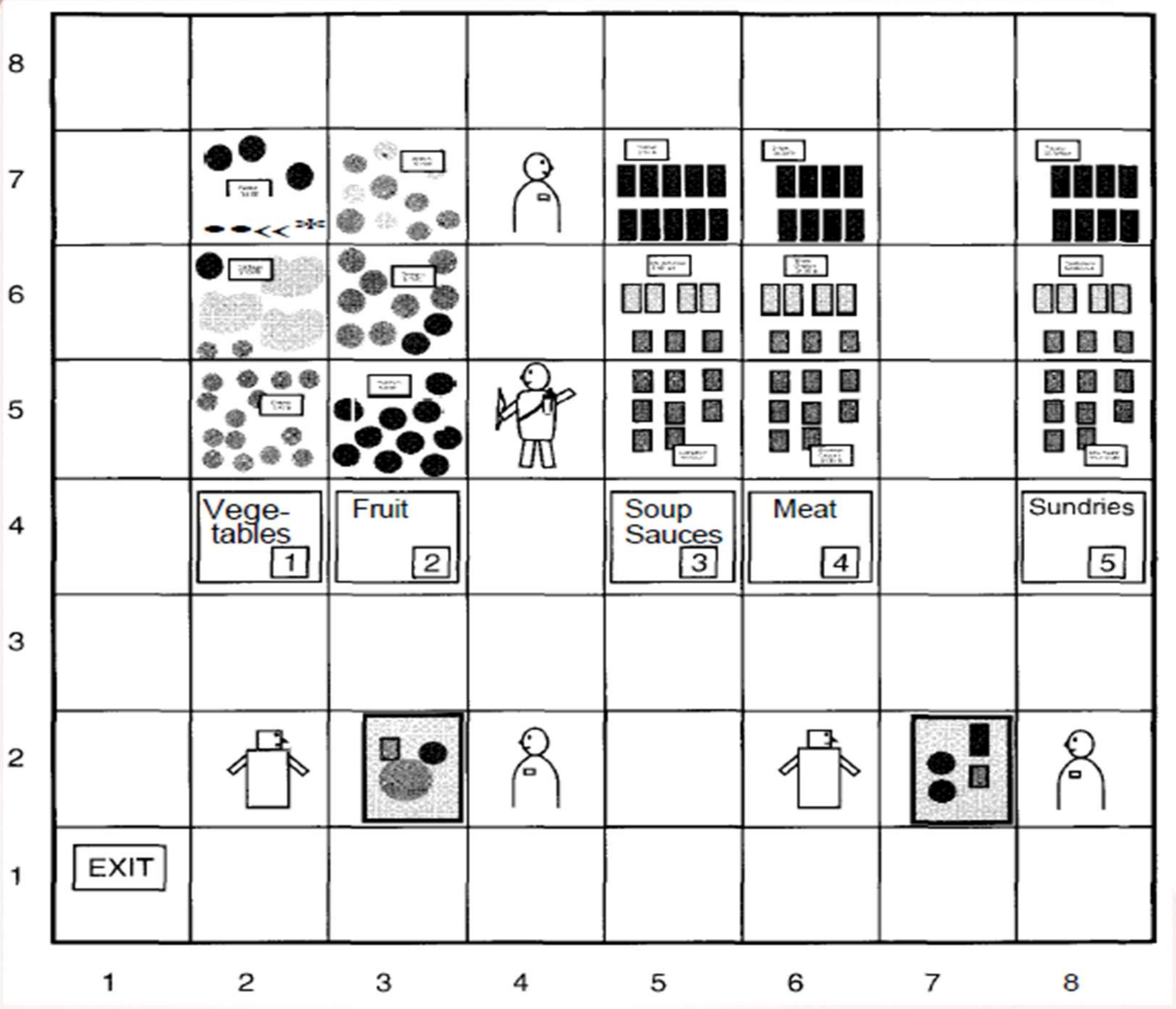
THE GROCERY SHOPPING WORLD

- The agent's **goal** initially will be to buy all the items on a shopping list
- The **environment** is the interior of a store, along with all the objects and people in it
- The store is represented by a grid of squares, with aisles separating rows of display cases
- At one end of the store are the checkout stands and their attendant clerks
- Other customers and store employees may be anywhere in the store
- The agent begins at the entrance, and must leave the store from the same square
- There is an EXIT sign there in case the agent forgets
- There are also signs marking the aisles, and smaller signs (readableonly when the camera zooms in) marking some (but not necessarily all) of the items for sale

THE GROCERY SHOPPING WORLD

- The first component of each description is its relative position with respect to the agent's position and orientation
For example, the relative position [-2,1] is the square two squares to the agent's left and one square ahead
- The second component is the size of the object, given as the average diameter of the object in meters
- Next is the color of the object, given as a symbol (red, green, yellow, orange, ...), followed by the object's shape (flat, round, square, ...)

THE GROCERY SHOPPING WORLD



Organizing knowledge

- **Menu Planning:** The agent will need to know how to modify the shopping list when the store is out of stock of an item
- **Navigating:** As in the wumpus world, the agent will need to understand the effect of movement actions and create an internal map of the world
- **Gathering:** The agent must be able to find and gather the items it wants. Part of this involves inducing objects from percepts: the agent will need recognition rules to infer that a red roughly spherical object about three inches in diameter could be a tomato
- **Communicating:** The agent should be able to ask questions when there is something it cannot find out on its own
- **Paying:** Even a shy agent that prefers not to ask questions will need enough interagent skills to be able to pay the checkout clerk

Menu-Planning

- To make these inferences, an agent needs to understand that the items on a shopping list fit together to form one or more composite objects known as **dishes,** that the dishes go together to form composite objects known as **meals**, and that an object can be recognized by its components
- Make two classes of inference.
- First, from a list of parts it should induce the composite object that these parts make up
- Second, the agent should be able to decide how to replace an unavailable part to complete the intended composite object

Menu-Planning

- The first step is to convert a shopping list—a list of words—into a parts list—a list of categories

Referent("tomatoes",Tomatoes)

Referent(" onions", Onions)....

- The next step is to describe objects in terms of their required and optional parts

$\forall r,w$ RequiredParts(r,w) $\Rightarrow$ Mp p $\in$ r $\Rightarrow$ RequiredPart(p,w)

$\forall o,w$ OptionalParts(o,w) $\Rightarrow$ $\forall p$ p $\in$ o $\Rightarrow$ OptionalPart(p,w)

$\forall r,w$ RequiredPart(r, w) V c $\in$ e w $\Rightarrow$ $\exists i$ i $\in$ r Λ PartOf(i, c)

$\forall o,w$ OptionalPart(o,w) $\Rightarrow$ $\exists c$ c$\in$w $\Rightarrow$ $\exists i$ i $\in$ o A PartOf(o, c)

Menu-Planning

- The next step is to describe meals and dishes in terms of their parts:

  RequiredParts({MainCourses}, Meals)

  Optional Parts({FirstCourses, SideDishes, Salads,Desserts,...},Meals)

  RequiredParts({Lettuce, Dressing}, GreenSalads)

  Optional Part s({Tomatoes, Cucumbers, Peppers, Carrots,...}, GreenSalads)

  RequiredParts({Pasta, BologneseSauce}, PastaBolognese,)

  Optional Parts({ GratedCheese], PastaBolognese)

  RequiredParts({Onions, OliveOil, Butter, Celery, Carrots, GroundBeef, Salt, WhiteWines,Milk, TomatoStuff}, BologneseSauce)

Menu-Planning

- Then we need taxonomic information for dishes and foods:

    GreenSalads C *Salads*

    Salads C *Dishes*

    PastaBolognese C *FirstCourses*

    FirstCourses C *Dishes*

    Tomatoes C *TomatoStuff*

    CannedTomatoes C *TomatoStuff*

    Tagliatelle C *Pasta*

- Now we want to be able to determine what dishes can be made from the shopping list "tomatoes, yellow onions, celery, a carrot, ground beef, milk, white wine, tagliatelle."

Navigating

- A shopping agent should know that supermarkets are arranged into aisles, that aisles have signs describing their contents (in rough terms), and that objects that are near each other in the taxonomic hierarchy are likely to be near each other in physical space
- A typical navigation problem is to locate the tomatoes. The following strategy is usually believed to work:
  - If the agent knows the location of the tomatoes from a previous visit, calculate a path to that spot from the current location
  - Otherwise, if the location of the vegetable aisle is known, plan a path there
  - Otherwise, move along the front of the store until a sign for the vegetable aisle is spotted
  - If none of these work, wander about and find someone to ask where the tomatoes are
  - Once the vegetable aisle is found, move down the aisle with the camera zoomed out, looking for something red. When spotted, zoom in to see if they are in fact tomatoes

Gathering

- Once in the right aisle, the agent still needs to find the items on its list
- The shopping agent can use the following classification rules:
- If only one known category matches a percept, assume the object is a member of that category
- If a percept matches several categories, but there is a sign nearby that identifies one of them, assume the object is a member of that category
- If there is an aisle sign that identifies one category (or one supercategory), assume the object is of that category

Gathering

- To implement this, we start with a set of *causal* rules for percepts:

  $\forall x$ *x* G *Tomatoes* $\Rightarrow$ *SurfaceColor(x, Red)*

  $\forall x$ x $\in$ Oranges $\Rightarrow$ *SurfaceColor(x, Orange)*

  $\forall x$ *x* $\in$ *Apples* $\Rightarrow$ *SurfaceColor(x, Red)* V *SurfaceColor(x, Green)*

  $\forall x$ x $\in$ Tomatoes $\Rightarrow$ *Shape(x, Round)*

  $\forall x$ *x* G *Oranges* $\Rightarrow$ Shape(x, Round)

  .

  .

  $\forall x$ *SurfaceColor(x, c)* Λ *Visible(x)* $\Rightarrow$ *CausesColorPercept(x, c)*

  $\forall x$ *Shape(x, s)* Λ *Visible(x)* $\Rightarrow$ *CausesShapePercept(x, s)*

Communicating

- If a word appears on an aisle's sign, then members of the category that the word refers to will be located in that aisle
∀a (a e Aisles Λ ∃s,w SignOf(s, a) Λ w G Words(s)) ⇒
∃x, c Referent(w, c) Λ x G c Λ At(x, a)

- If a word appears on a small sign, then items of that category will be located nearby.
∀s,w, l (s G Signs Λ Size(s) < Meters(3) Λ w G Words(s) Λ At(s, l)) ⇒
∃x, c Referent(w, c) Λ x G c Λ At(x, AreaAround(l))

Paying

- Agent needs to know typical fair prices for items, for example:
$\forall g\ g \in Typical(GroundBeef) \wedge Weight(g) = Pounds(l)$
$\Rightarrow \$(1) < FairPrice(g) < \$(2)$

- The agent should know that total price is roughly proportional to quantity, but that often discounts are given for buying larger sizes. The following rule says that this discount can be up to 50%:
$\forall q, c, w, p\ q\ G\ c \wedge Weight(q) — w \wedge Price(q) = p$
$\Rightarrow \forall m, q2\ m > 1 \wedge q2 \in c \wedge Weight(q2) = m \times w$
$\Rightarrow (1 + (m-1)/2) \times p < FairPrice(q2) < m \times p$

- *Agent should know that it is a bad deal to pay more than the fair price for an item, and that buying anything that is a bad deal is a bad action:*
$\forall i\ Price(i) > FairPrice(i) \Rightarrow BadDeal(i)$
$\forall i\ BadDeal(i) \Rightarrow Ma\ Bad(Buy(a, i))$

Paying

- Buying events belong to the category Buy(b,x,s,p)—buyer b buying object x from seller s for price p
- The preconditions include the fact that p is the price of the object x; that b has at least that much money in the form of one or more monetary instruments; and that s owns x
- The event includes a monetary exchange that results in a net gain of p for s, and finally b owns x
- It is bad form to exit a shop while carrying something that the shop owns
$\forall a, x,s,i\ s \in \text{Shops}\ \Lambda\ T(\text{Carrying}(a, x)\ \Lambda\ \text{At}(x, s)\ \Lambda\ \text{Owns}(s, x))$
$\Rightarrow T(\text{Bad}(\text{Exit}(a)), i)$

Summary

- The process of representing knowledge of a domain goes through several stages. The first, informal stage involves deciding what kinds of objects and relations need to be represented (the ontology). Then a vocabulary is selected, and used to encode general knowledge of the domain. After encoding specific problem instances, automated inference procedures can be used to solve them.

- Good representations eliminate irrelevant detail, capture relevant distinctions, and express knowledge at the most general level possible.

- Constructing knowledge-based systems has advantages over programming: the knowledge engineer has to concentrate only on what's true about the domain, rather than on solving the problems and encoding the solution process; the same knowledge can often be used in several ways; debugging knowledge is often simpler than debugging programs.

Summary

- Special-purpose ontologies, such as the one constructed for the circuits domain, can be effective within the domain but often need to be generalized to broaden their coverage.
- A general-purpose ontology needs to cover a wide variety of knowledge, and should be capable in principle of handling any domain.
- Presented a general ontology based around categories and the event calculus. Covered structured objects, time and space, change, processes, substances, and beliefs.
- Presented a detailed analysis of the shopping domain, exercising the general ontology and showing how the domain knowledge can be used by a shopping agent.

Thank You!