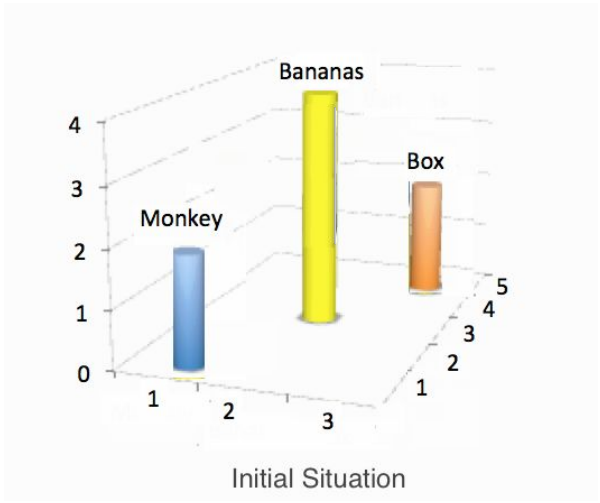
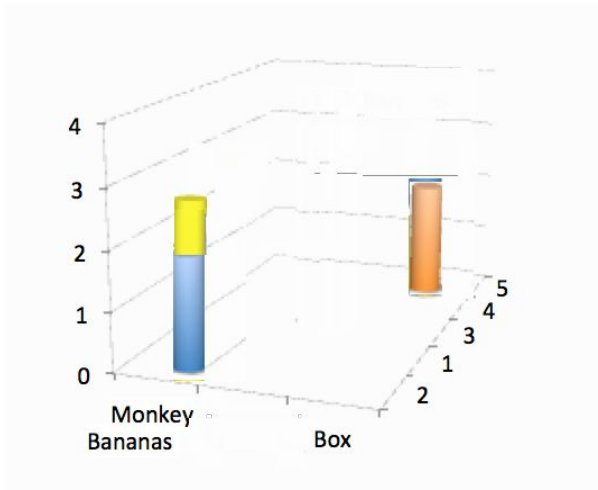
Traci Fairchild
tfairchild3
KBAI Mid-Term

Q1 (25 points): In the classic Monkey & Bananas problem, a monkey is faced with the problem of reaching bananas hanging from the ceiling. The bananas are too high for the monkey to reach. But a box is available in the room that will enable the monkey to reach the bananas if he climbs on it. Initially the monkey is at location A, bananas at B, and the box at C. The bananas are at height Y, and the monkey and the box each have height X such that if the monkey climbs on the box, it too will be at height Y. So the monkey pushes the box to just under the bananas, climbs the box, and gets the bananas. Intelligent monkey!
In the modern variation of the problem, some scientists are observing the monkey and the monkey is observing the scientists. As scientists take a break to have some tea, the monkey decides to fool the scientists. He pushes the box to just under the bananas, climbs up the box, gets the bananas, climbs down the box, pushes the box to its initial position, and resumes its repose. Very intelligent monkey!

Invent operators Walk, Push, Climb-Up, Climb-Down, and Get for the above problem. Show how the monkey may used means-ends analysis to form a plan in the second scenario.

<table>
<tr><td colspan="2">Initial situation
shows the initial location of the monkey, the box and the bananas</td></tr>
<tr><td> 

Initial Situation</td><td>ONSQUARE(monkey,1,1)
ONSQUARE(box,3,5)
ONSQUARE(bananas,2,3)
TEA(scientists)</td></tr>
</table>

Goal situation shows the goal location of the monkey, the box and the bananas	
	ONSQUARE(monkey,1,1) ONSQUARE(box,3,5) ONSQUARE(bananas,1,1) TEA(scientists)

The goal situation can be reached with the operator descriptions below:

Operator Walk Monkey walks from square (x,y) to an adjacent square (xx,yy)	
If:	ONSQUARE(monkey,x,y) NEAR(monkey,xx,yy) ISEMPTY(xx,yy)
Add List:	ONSQUARE(monkey,xx,yy) CLEAROBJ(monkey,x,y) NEAR(monkey,x,y) ISEMPTY(x,y)
Delete list:	ONSQUARE(monkey,x,y) CLEAROBJ(monkey,xx,yy) NEAR(monkey,xx,yy) ISEMPTY(xx,yy)

Operator Push Monkey on square(x,y) pushes box on square(xx,yy) to an adjacent square(xxx,yyy) and the monkey moves to square(xx,yy)	
If:	ONSQUARE(monkey,x,y) ONSQUARE(box,xx,yy) NEAR(monkey,xx,yy) NEAR(box,xxx,yyy) ISEMPTY(xxx,yyy)
Add List:	ISEMPTY(x,y) CLEAROBJ(monkey,x,y) CLEAROBJ(box,xx,yy) ONSQUARE(monkey,xx,yy) ONSQUARE(box,xxx,yyy) NEAR(monkey,xxx,yyy)
Delete list:	CLEAROBJ(monkey,xx,yy) CLEAROBJ(box,xxx,yyy) ONSQUARE(monkey,x,y) ONSQUARE(box,xx,yy) ISEMPTY(xxx,yyy) NEAR(monkey,xx,yy) NEAR(box,xxx,yyy)

Operator Climb-Up Monkey on square(x,y) climbs up on box located on square(xx,yy)	
If:	ONSQUARE(monkey,x,y) ONSQUARE(box,xx,yy) NEAR(monkey,xx,yy)
Add List:	ONSQUARE(monkey,xx,yy) ONSQUARE(box,xx,yy) CLEAROBJ(monkey,x,y) NEAR(monkey,x,y) ISEMPTY(x,y)
Delete list:	ONSQUARE(monkey,x,y) ONSQUARE(box,xx,yy) NEAR(monkey,xx,yy)

Operator Climb-Down Monkey on box which is on square(xx,yy) climbs down off box onto square(x,y)	
If:	ONSQUARE(monkey,xx,yy) ONSQUARE(box,xx,yy) NEAR(monkey,x,y) ISEMPTY(x,y)
Add List:	ONSQUARE(monkey,x,y) ONSQUARE(box,xx,yy) NEAR(monkey,xx,yy)
Delete list:	ONSQUARE(monkey,xx,yy) ONSQUARE(box,xx,yy) NEAR(monkey,xx,yy)

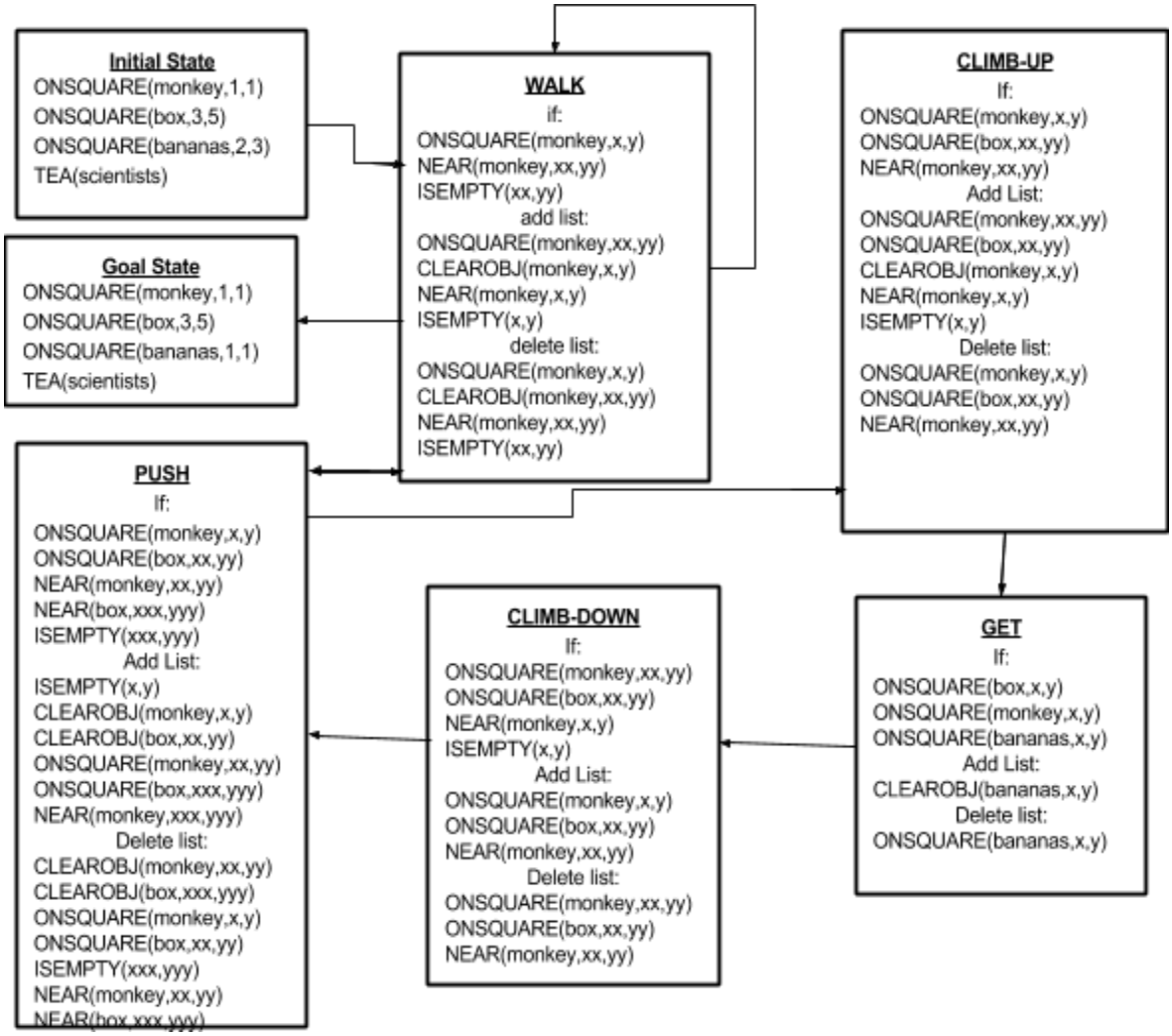
Operator Get Monkey on box, directly below bananas gets the bananas	
If:	ONSQUARE(box,x,y) ONSQUARE(monkey,x,y) ONSQUARE(bananas,x,y)
Add List:	CLEAROBJ(bananas,x,y)
Delete list:	ONSQUARE(bananas,x,y)

The monkey could plan the scenario of retrieving the bananas when the scientists are on their tea break by using the Means-End Analysis in its planning. The purpose of MEA in planning is to move from the current state to an intermediate state which is closer to the goal state. The operators defined above would be selected appropriately to bring the monkey closer to the goal state. Put differently, the goal of each choice is to reduce the difference between the monkey's current state (where he is, where the box is, and where the bananas are) and the goal state, which is standing on the box directly under the bananas and grabbing them.

Before showing the planning design, we might benefit from a Difference Procedure Table which will help the monkey decide on which operator to use next, Walk, Push, Climb-Up, Climb-Down, and Get:

Description	Walk	Push	Climb-Up	Climb-Down	Get
Monkey is not near the box	✓				
Monkey is near the box		✓	✓		
Box is not underneath the bananas	✓	✓			
Box is underneath the bananas			✓	✓	✓

And now the plan that the monkey will use to get to the bananas



Using MEA, the monkey would take a step using the WALK operator. After this step, the monkey would consider the current state. It may be farther from the bananas, but it is closer to the box so it decides to proceed and not move back. Again, using the WALK operator, it takes another step. This time, the monkey moved farther away from the box, but closer to the bananas. The monkey decides that this has brought him further away from the goal state because he is smart enough to know that he needs to reach the box first. Therefore, the monkey decides to abandon its last move and go back. Again, it attempts to take another step in a different direction. This direction is towards and closer to the box, so it proceeds. The monkey repeats this logic until it finally reaches the box. The monkey has reached the box by abandoning the steps that took it further away and is now left with an optimal path to the box. It feels it is halfway to the goal state. Now it must push the box to directly underneath the bananas. In the same way, the monkey uses the PUSH operator, pushing the box from square to square. At each move, the monkey looks up and looks to see if it is underneath the bananas or closer to them. The monkey abandon's the moves that take it further from the bananas.

Finally, the monkey reaches the square that is directly underneath the bananas. It is time to use the CLIMB-UP operator. The monkey easily climbs the box. In MEA, the next step could be the CLIMB-DOWN operator, as the very design of MEA is to consider all possible combinations of movements. However, climbing down will not bring the monkey closer, but further away from the goal state. The monkey decides to throw this move out. The next move the monkey chooses is to grab the bananas using the GET operator. The monkey uses the CLIMB-DOWN operator. With the bananas in hand, the monkey repeats his previous steps in reverse order, drops the box back to it's original position, and continues to walk back to its original position. The monkey has reached the goal state.

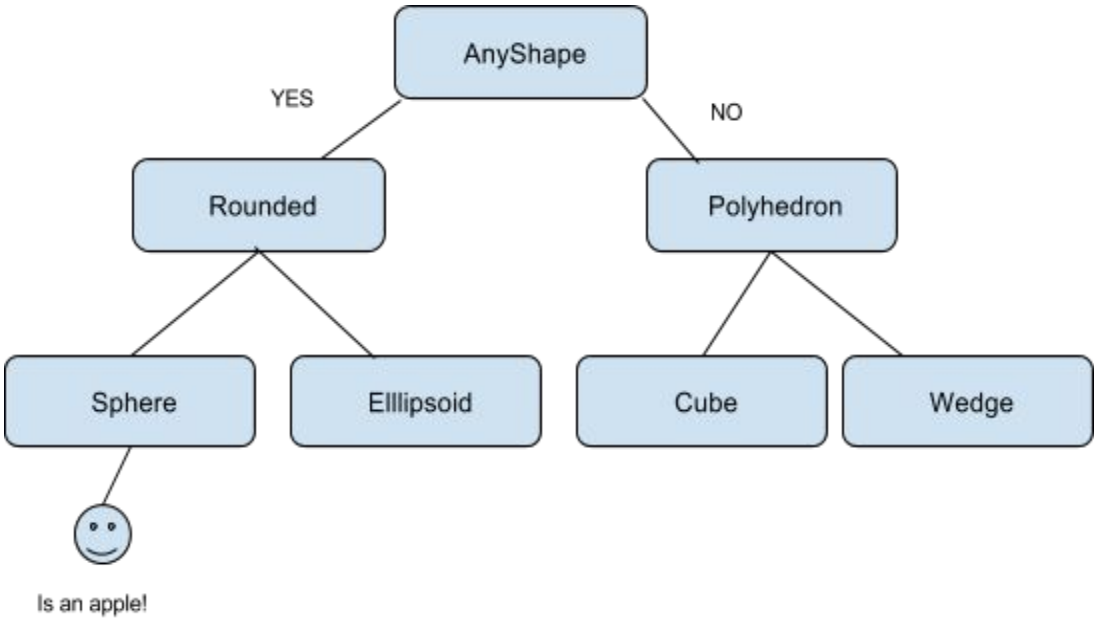
Q2 (25 points): You want to teach a Martian about the concept of an apple. You may assume that the Martian has some background knowledge:

- **(a) An object's color can be red, green, blue, yellow, white or black.**
- **(b) An object's weight is an integer.**
- **(c) An object's texture may be smooth, spotty, rough or wrinkly,**
- **(d) An object can take the shapes in the tree below.**

Design a sequence of examples for teaching the Martian about the concept of an apple. Show the evolving definition of the concept after each example.

This is an example of Incremental Concept Learning. Based on what the martian already knows about the world, we need to define and redefine our concept of an apple.

The first thing the martian is likely to notice about an object is its shape. The martian, therefore, may ask itself if this is the shape of an apple, and if so then it may believe the object is an apple:

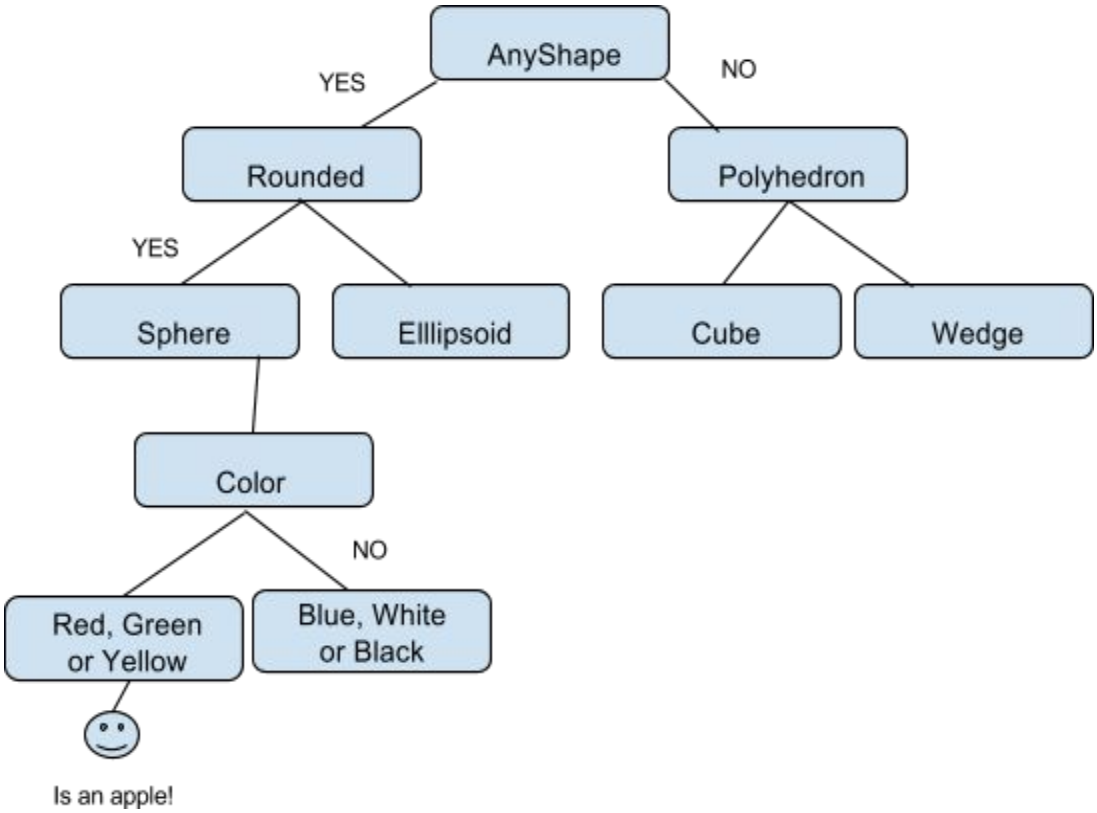


The martian may believe that it has found the expression of an apple. The NO expression above is an example of using the forbid-link heuristic in that the concept must not be present. In this example, it must not have any polyhedron shaped objects. The YES link above is referred to as the require-link heuristic, meaning the concept must be present. In the concept of an apple, the object must be rounded and spherical:

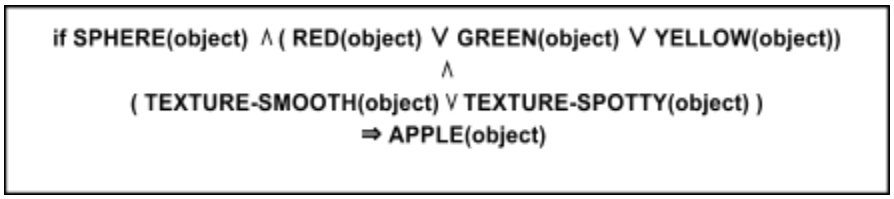
$$\text{if (ROUNDED(object)} \vee \text{if } \neg\text{POLYHEDRON(object))} \wedge \text{if SPHERE(object)} \Rightarrow \text{APPLE(object)}$$

So let us test out the concept of an apple above by giving the martian a softball. It is clearly a rounded, spherical object, yet it is not an apple. We therefore need to specialize our concept definition further. This is an example of a *Specialization to Require Features*, again, the require-link heuristic. So, given additional knowledge the martian has of color, we can say that an apple is RED, GREEN, or YELLOW and that it's color is not BLUE, WHITE OR BLACK. So now the martian believes our refined definition:

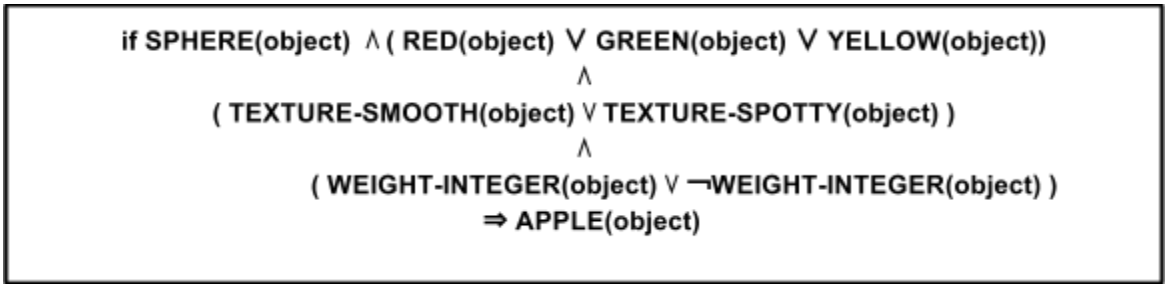
$$\text{if SPHERE(object)} \wedge (\text{RED(object)} \vee \text{GREEN(object)} \vee \text{YELLOW(object)}) \Rightarrow \text{APPLE(object)}$$



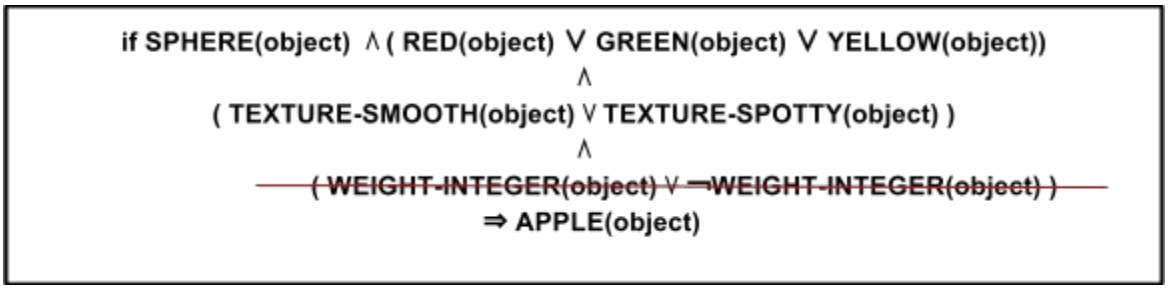
Let's give the martian a yellow tennis ball. Looking at our concept of an apple, is the tennis ball an example of this concept? No, a tennis ball is not an example of an apple, yet it fits our concept, so we need to specialize our concept even more.
You see how this is going, next we take more of what the martian understands and we try to add it to our concept of an apple. Rather than expanding on the diagram, I will add the additional logic expression:



The last piece of information that the martian understands is that an object's weight is an integer.



But the object's weight does not add to our concept of an apple since an apples weight can be an integer or a float. Therefore, let us drop this link in the concept. This is an example of *Generalization to Ignore Features* using the drop-link heuristic as discussed in class.



Q3. Consider, as an example, the seemingly simple but actually quite intricate task of preparing milk in a bottle for feeding a toddler. You warm some milk in the microwave, pour it in a clean bottle, put the cap on, and you are all set to feed the child. The robot observes your actions. You may even carefully demonstrate the actions to the robot.

Using the concepts and methods you have learned in the KBAI class, develop a computational technique for learning by imitation using the above scenario as an example. Show the computational processes and knowledge representations. Illustrate your answer by working through the example.

Cognitive agents learn from their environment. In the act of learning, they store items in their long term memory, just like humans do. Our robot may already have some procedural experiences of how to do things stored in memory, but at this point our Robot is fairly new and has never heated a bottle of milk before. Not only will this event be stored as an episodic event, it will also be stored in and retrieved from it's procedural knowledge database so that when it is asked to heat the bottle again, it will know how to do it. Of course, the robot came pre-packaged with a vast semantic knowledge database. Semantic memory refers to a portion of long-term memory that processes ideas and concepts that are not drawn from personal experience. Semantic memory includes things that are common knowledge such as the names of colors, the sounds of letters, the capitals of countries and other basic facts acquired over a lifetime.[1] Our robot was provided with a massive amount of information that it can use as it observes me preparing heated milk in a bottle.

It is important to put myself, as a human, in the place of the robot. Although the robot has sensors, actuators and information at its disposal, it is not sentient. It cannot understand things like humans do. Instead it processes its environment by reading sensor data that is constantly incoming. This information, in the form of vast amounts of measurements is what the robot uses to calculate, perform, store, retrieve and repeat a particular process that it has "observed".

When presented with this new task of preparing heated milk in a bottle, I define the initial goal state for the robot, that is, no bottle with heated milk ready. The goal state would be a bottle with heated milk ready. The constraints of this problem are that the container must be a bottle, the bottle must have a lid, the bottle must be clean, the bottle must be filled with milk, the milk must be warm.

Initial State: No bottle filled with heated milk

Goal State: A bottle filled with heated milk

Constraints:

1. must be in a bottle
2. bottle must be clean
3. bottle must have a lid
4. must be filled with milk
5. milk must be warm

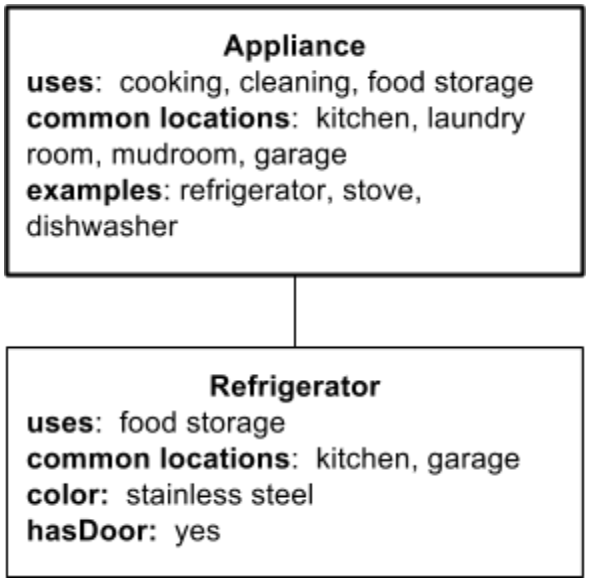
With this initial problem information, the robot chooses to forego means-ends analysis (MEA) which is useful for broad searches in problem solving. Instead, this problem is well formed and the solution will be shown to the robot. Because of this, it is best to proceed with the problem reduction method. Problem reduction involves decomposing the problem into simpler problems, such as the following:

1. goto refrigerator
2. get milk
3. goto cabinet
4. get container
5. pour milk into container
6. goto microwave
7. heat milk in microwave
8. goto cabinet
9. get bottle
10. pour milk into bottle
11. close bottle

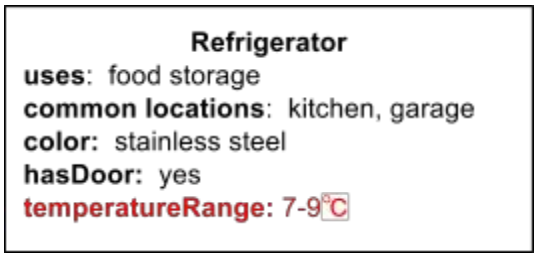
The operators for these problem subsets could look something like this:

- goto(refrigerator)
- goto(cabinet)
- goto(counter)
- goto(microwave)
- select(milk)
- select(container)
- select(measuring-cup)
- select(bottle)
- put(milk, counter)
- operate(microwave)
- open(cabinet)
- open(refrigerator)
- open(microwave)
- open(bottle)
- open(milk)
- pour(milk, measuring-cup)
- put(milk, refrigerator)
- put(measuring-cup,sink)
- put(measuring-cup, microwave)
- close(refrigerator)
- close(cabinet)
- close(bottle)

Let us assume that the robot already has a stored spatial map of the home it is in. And let us suppose that the robot was one of the cheaper models from last year and so it does not have up to date information on the latest appliances models. I first begin by showing the robot around the kitchen, pointing out cabinets and appliances so that it may learn. The robot has a concept of some appliances already:



But all the appliances are brand new in this home and do not exactly fit the definitions the robot has retrieved from it's own memory. It must apply what it knows about appliances and modify its concept of what it has retrieved to find where the refrigerator actually is. The robot believes that a refrigerator is a stainless steel appliance with a door so it moves to towards the stove believing it has found the refrigerator. I inform the robot that it is not a refrigerator. Indeed, the robot opens the stove door and measures the temperature, it is not cold inside, nor does it have food in there. So the robot must adjust its concept of a refrigerator by requiring specific details to it's concept of a refrigerator. It must add a require-link to it's concept. It chooses to add that a refrigerator is required to be cold on the inside.



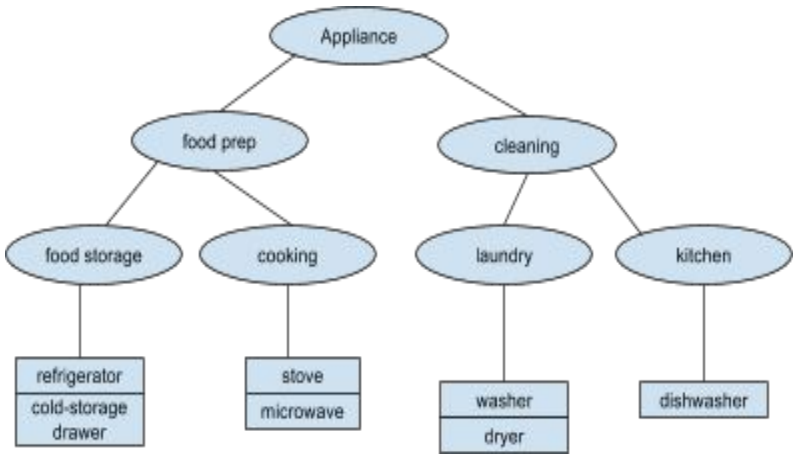
Finally I show the robot the actual refrigerator, which is black. The robot processes that the appliance is not stainless steel as anticipated so again it incrementally adjusts its concept to drop the requirement of color, as it now knows that color is not necessary to be a positive example of a refrigerator. Therefore, it applies the drop-link heuristic to remove color from its concept.

Refrigerator
uses: food storage
common locations: kitchen, garage
~~**color:** stainless steel~~
hasDoor: yes
temperatureRange: 7-9℃

I inform the robot that this is a positive example of a refrigerator, however, the robot chose the wrong door. The robot opens the left side of the refrigerator, which is the freezer side. The robot again adjusts its concept and adds a forbid-link that if the temperature inside is below the ideal refrigerator temperature and closer to a 0℃ then it is not a refrigerator, it is a freezer.

Refrigerator
uses: food storage
common locations: kitchen, garage
~~**color:** stainless steel~~
hasDoor: yes
temperatureRange: 7-9℃
temperatureNotBelow: 5℃

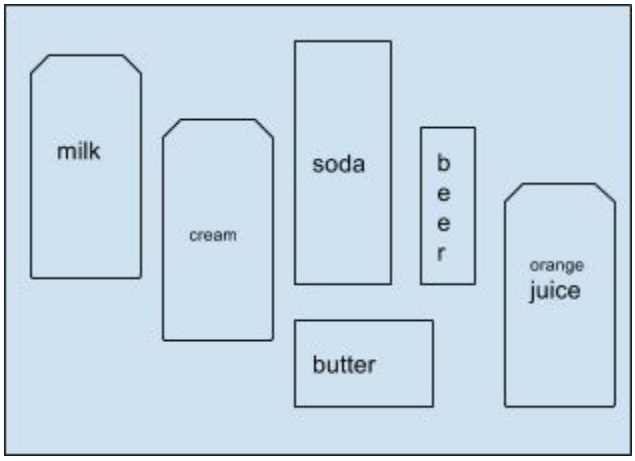
As the robot is learning about the appliances in the kitchen, it comes across a cold-storage drawer on the center island. This drawer is like a refrigerator and is as cold as a refrigerator. I inform the robot that it is a type of refrigerator. The robot is equipped with an internal climb-tree to help broaden or generalize it's concept of a refrigerator to span multiple instances each playing the same role. Using this climb tree the robot will have to adjust its concept of a refrigerator using incremental concept learning.



Now that the robot is at the refrigerator, I let it try to identify the milk using its classification system it has in memory of foods. It knows that milk is a perishable food, is a white liquid and is stored in the refrigerator.

object: milk
classification: food
state: liquid
color: white
perishable: yes
foodStorageTemp: 7-9C
container: plastic bottle

In the factory, the robot was programmed that milk was in a gallon, plastic bottle. But the milk here is not in a plastic bottle. It is in a cardboard carton - a milk carton. Using a nearest-neighbor algorithm of what it knows of common liquids, the robot tries to identify the carton of milk.



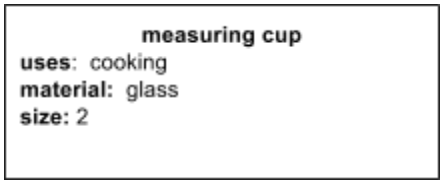
It selects a carton similar in size. Reading the label it learns that it is orange juice and not milk. The robot now knows that orange juice is often stored in a similar container. Looking for a white liquid, the robot finds the cream. Will the robot adapt the recipe and heat the cream instead of milk for the baby? No! The robot knows from its classification and semantic knowledge that cream is no substitute for milk when feeding a child. It therefore abandons the choice of heating cream to appease the child and continues looking for the milk using, once again, the nearest-neighbor algorithm. The process of retrieving knowledge from memory and evaluating it for adaptation or for abandonment is called Case Based Reasoning and that is what our robot just did. The robot successfully finds the milk, scans its nutrition label to verify and brings it to the counter to prepare it.

Next I go to the cupboard and get a glass measuring cup. By the time I bring it to the counter the robot has retrieved from it's memory a set of frames it has on containers:

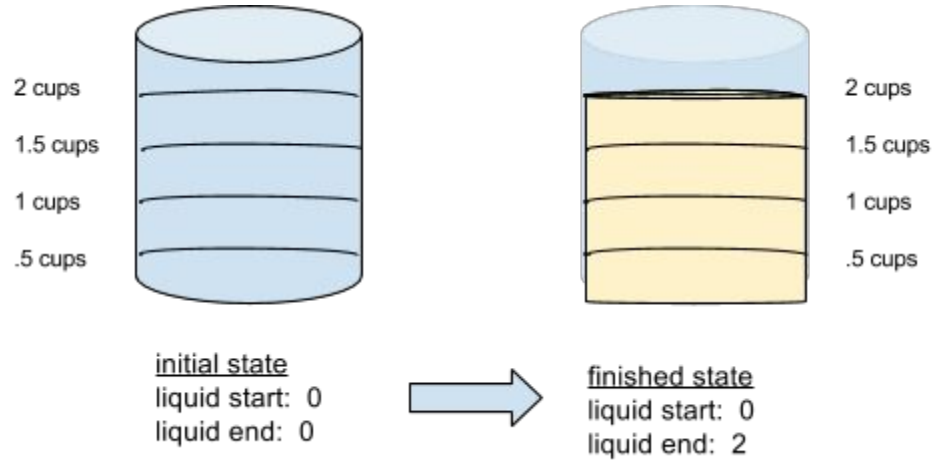
Container
uses: storage, heating, cooking
common locations: cabinet, cupboard, shelf

Just like the real world, the robot's frame representation of a container, usually stored in memory as a class or object, will contain variables and data and provide default values. These

values allow the robot to pass them down (as in inheritance) to a child structure of the same object class.



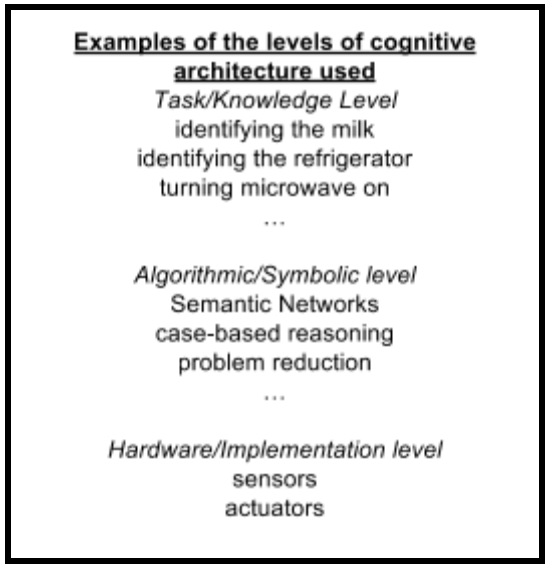
Next I let the robot observe me pouring the milk. I show the empty measuring cup as an initial state for the robot to record. I then pour milk into it and again show the robot. Since the robot can easily observe differences between states, it compares the initial state to the full state and knows that I just filled it up with 16 oz of milk. The robot can now infer that the bottle for the child will be approximately 16 oz in size.



I let the robot take the container and put it into the microwave. As the robot does this, it observes that the microwave keypad is a standard keypad that it is already familiar with. In fact, the microwave came from the same factory as the robot and they are good friends. The robot knows that milk boils at approximately 100 ℃ but the robot knows this will be too hot to serve. I show the robot to set the microwave on 70% power and cook for 15 seconds, open the door and stir. I repeat the process until the robot and I detect steam rising from the milk. The robot records this in its procedural memory for future retrieval.

Now the milk is heated, it is time to select a container. But there are many to choose from. The robot generates a scenario for each container it sees, and easily discards the solutions for containers that are not bottles at all. The robot is left with 3 remaining states. One state is a bottle with no cap, the robot discards this. The other state is a bottle that already has something in it, possibly a dirty container. The robot discards this state. The state remaining is a clean bottle with a cap. The robot chooses this. The process of generating and pruning possible solutions for a task is called the generate and test method and this is what the robot just did.

Using its arms and actuators, the robot carefully pours the heated milk into the bottle, seals it and presents it to the anxious child. The robot has now used all these levels of cognitive architecture in preparing the bottle of milk.



and has already stored this specific event into it's memory so that it is able to repeat it and learn from it. It can also be cloud-based stored so that other robots can retrieve and perform the same tasks with adaptation when necessary.

1. Live Science (January 29, 2014) Semantic Memory: Definition & Examples. Available: http://www.livescience.com/42920-semantic-memory.html [Accessed 01 October 2015]