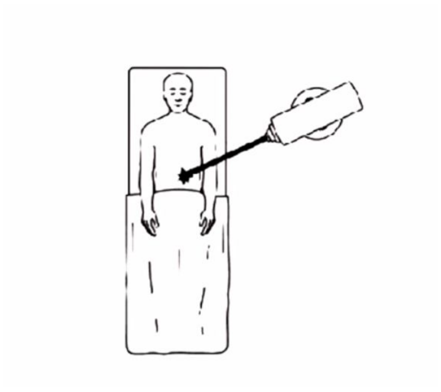
Final Exam KBAI - Vinodh Kumar Sunkara, Microsoft

Question 7: *Demonstrate the process of analogical reasoning. Take an existing system, and by analogy extend its design and solution to a new problem. In doing so, walk through all the steps of analogical reasoning. Finally, contrast this process with how you might address this problem by case-based reasoning.*

Sol. Analogical reasoning is used to characterize methods that solve new problems based on past cases from a different domain, while typical case-based methods focus on indexing and matching strategies for single-domain cases. Analogical reasoning allows us to look at new problems in terms of family of problems. It also allows us to transfer knowledge from a family of problems to new problem. It consists of five major phases: Retrieval, Mapping, Transfer, Evaluation and Storage. In analogical reasoning, the target problem and source case need not be from same domain. When they are not from same domain, we can't just take the source case and adapt it like in case-based reasoning.

Let's consider a problem of laser beam applied on a patient to treat tumors. The beam is strong enough to kill tumors but it'll also kill the healthy tissue in between. We've to find a solution to this problem to apply laser beam in a way that healthy tissue is unaffected killing the tumors. Now let's take a case of rebel army attacking king in palace. The four pathways into palace are mined, and a big group of soldiers passing through it would let the mines go off and kill everyone. The solution here is to break the soldiers into four small groups and let them pass through four paths from different directions to enter palace and attack king. The mines don't go off with small groups of soldiers walking on.



In order to solve the laser patient problem, we first retrieve the above king-soldier case from storage. We can use either superficial similarity or deep similarity for retrieving similar case. In our example, we use deep similarity which deals with relationship between objects or relationship between relationships. There are three types of similarities - semantic (conceptual similarity between target and source), pragmatic (Similarity of external factors such as goals) and structural (representational structures similarity). In our example, we use pragmatic similarity which is the similarity between goal of capturing king and destroying the tumor. We'll then map the target problem to source case. In this case, laser resource is mapped to soldiers and the goal tumor in source case is mapped to king in the problem. Then we'll abstract the relationship from source case and transfer to the target problem. In our example, relationship is to break resource into smaller chunks and release on to goal. We'll evaluate the proposed solution by simulating the solution. Once it's evaluated that laser beam can destroy tumor without affecting healthy tissue, the target problem and proposed solution are stored as a new case used in future for solving another problem.

We can solve the above problem using case based reasoning. We have the cases from same domain in case based reasoning. The steps in case based reasoning are Retrieval, Adaptation, Evaluation and Storage. The target problem is solved by tweaking solutions of similar previously encountered problems. Let's consider the source case in above example as target problem here. The target problem is that soldiers have to find a way to attack king while paths between are mined. Let's take a source case where soldiers have to cross a river to kill the king on other bank. There are four bridges to cross the river. All bridges are mined. Soldiers divide into four groups to cross the bridge through four different bridges and attack the king instead of everyone going through one of the four bridges. This is similar to target problem and case is retrieved from storage. This solution in soldiers-bridgesToBank-king case is adapted to the target problem of soldiers-pathsToPalace-king. Instead of four bridges, the four paths from different directions to palace are used and rest all remains the same in solution adapted. The proposed solution is evaluated by simulating it. If the king is attacked without putting off the mines, it's a valid solution. The target problem and adapted solution is stored as new case in storage. It can be stored by index or using discrimination tree. Discrimination tree can be on soldiers count check, mines on paths condition and four/multiple paths available to goal. After solving target problem, the tree is appended by a new condition on paths from different direction to goal as in paths to palace in target problem vs four parallel paths to goal as in source case to bank across river.

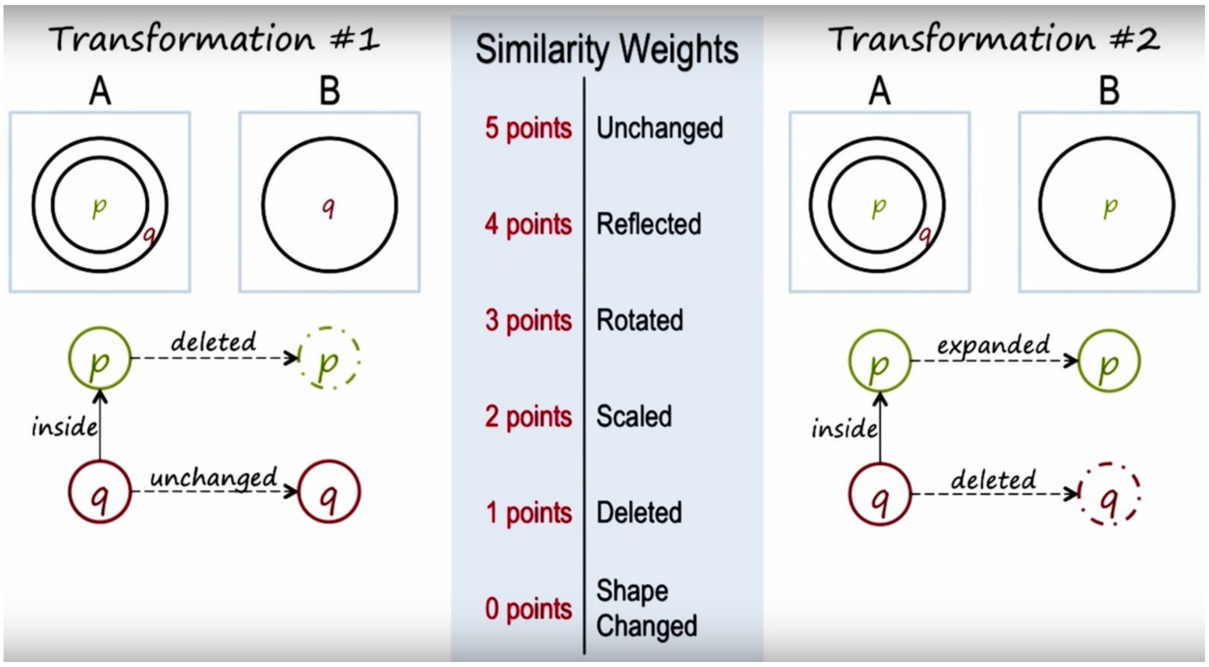
Question 3: *One of the challenges in solving Raven's Progressive Matrices is to determine which objects correspond with each other between figures. This problem is especially relevant to agents which reason verbally. Please describe, at a general level, a solution to the correspondence problem. Identify both a knowledge representation and problem solving strategy that would combine to feasibly solve the problem.*

Sol: It's really challenging to determine which objects correspond with each other between figures in Raven's Progressive matrices. We can find the correspondence between objects by framing a transformation between the knowledge representations of two figures. It gets challenging when there are multiple transformations and we've to pick the valid one. We would need to use a ranking schema on transformations to pick the valid one. We'll use the

Semantic Networks knowledge representation to illustrate the problem solving strategy for correspondence problem. Semantic networks has lexons/nodes, structure/links and semantic/labels to capture the knowledge representation.

We have labels associated with each object in the figure. We have same label for objects on other figure with same properties. For example, we can use 'shape' to assign label names. All circles are marked as 'x' in our Semantic networks knowledge representation. All the objects become nodes with above labels in the semantic networks. The nodes are connected by directional links to capture relationship between the objects within a figure. The links are also labelled appropriately to denote relationship between the objects in a figure. For example, 'inside' and 'above' are two relations for objects to have between each other within a figure. Once the knowledge from each figure is represented using labelled nodes and links, we've to mark the transformation between the two knowledge representations in order the capture the relationship between the two figures. We'll have links between knowledge representations of two figures to capture the transformation. For nodes with same shape, size etc. properties, it's straightforward link with 'unchanged' label whereas any changes happened between nodes are captured using links with different labels like 'expanded', 'deleted' etc. The similarity between transformation of knowledge representations for a pair of figures can be compared with transformation between another pair to confirm the stronger correspondence between objects and relationships.

Let's consider an example where object correspondence is challenging.



For above example, there are two possible transformations. The inner circle expands and outer circle is deleted in Transformation 2. The outer circle is unchanged and inner circle is deleted in Transformation 1. It's challenging to rate these transformations to pick the most appropriate one since both are valid transformations. We use the strategy of similarity weights to derive right correspondence between objects. We assign similarity weights on transformation link labels and pick the transformation with higher similarity score as the most appropriate one.

The similarity score of transformation 1 is: +1 (p-deleted) +5 (q-unchanged) = 6 points
The similarity score of transformation 2 is: +2 (p-expanded) +1 (q-deleted) = 3 points

Transformation 1 with higher similarity score is picked as the most appropriate for this pair of figures. The same transformation can be applied to one of the figures to find the missing figure in another pair.

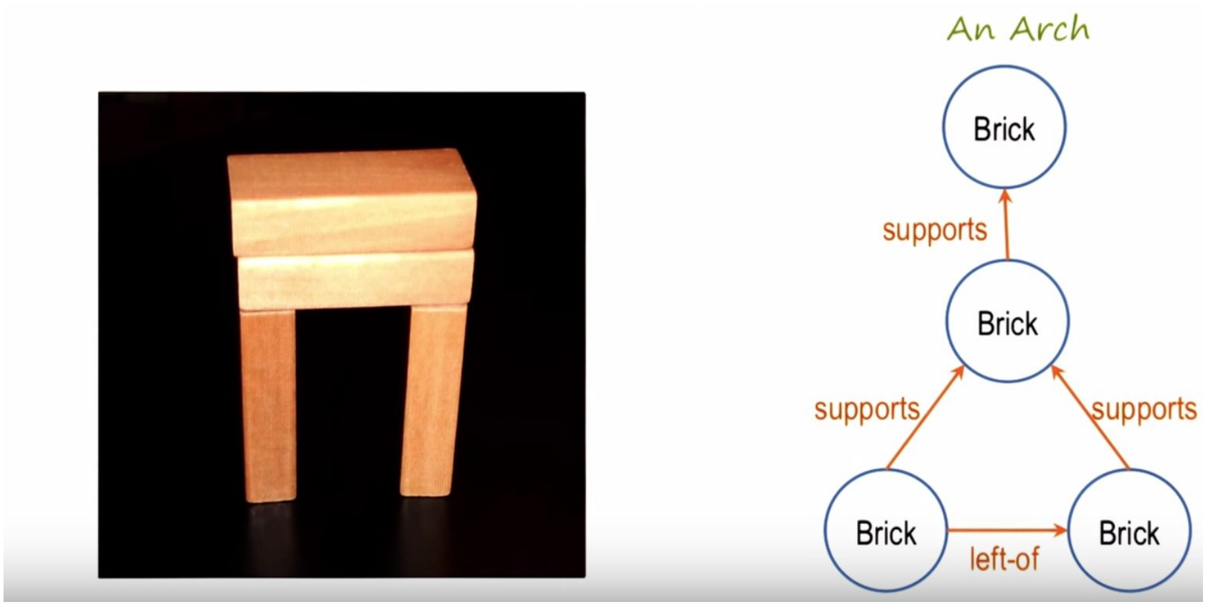
Question 5: *Incremental concept learning and version spaces are two methods that agents may use to learn from a small number of examples analyzed one at a time. In our lessons, we discussed incremental concept learning in terms of figures of blocks, and we discussed version spaces in terms of identifying the cause of an allergic reaction or food poisoning. Take one of these examples -- either identifying arrangements of blocks or identifying the cause of allergic reactions -- and attempt to resolve it using the other method. Use this attempt to compare and contrast the two methods: what are the strengths and weaknesses of each method? What kinds of problems are each method better suited to solve?*

Sol:

Version spaces is a technique for learning concepts incrementally. This technique is going to learn from a small set of examples one at a time. Version spaces technique of learning concepts incrementally will always have a specific model and a general model. As a new example comes along, we ask ourselves if it's a positive example or a negative example. If it's a positive example, then we generalize the specific model. If it's a negative example, we specialize the general model. The most specific model would match exactly one thing while the most general model matches everything. We'll start with most specific model on one side and the most general model on other, keep generalizing and specializing the models respectively as the positive/negative examples come in. Some of the generalizations and specializations we create may not match the current data and we prune them out. As we prune on both sides, the two pathways merge depending on the ordering of examples. When they merge, we've a solution which is right generalization of concept for the given examples.

In case of incremental concept learning, we make necessary changes to our current concept with every new example. If the new example is a positive case and concept already covers new example, we keep it unchanged. If the new example is positive and current concept doesn’t cover it, we must generalize the concept to include the new example. If the new example is negative case and fits in current definition of concept, we’ve to specialize our current concept. If the example is negative and current concept doesn’t cover it, we keep the concept unchanged. Ordering of the examples is important in incremental concept learning. We always want our first example to be a positive example to start with, and eventually generalize or specialize the concept in further revisions in case of positive and negative examples respectively. Incremental concept learning is very useful when there is a teacher available who gives the examples in right order. We want the set of examples to include both positive and negative sets so that we can do generalization and specialization in incremental concept learning. We also want the new example to differ from current concept in only one feature to revise concept effectively under incremental concept learning. We also need some background knowledge that helps in generalizing the concept. So, the availability of teacher to order examples and background knowledge guides the agent in the extent of concept generalization under incremental concept learning, while version spaces technique allows the agent to converge to right generalization irrespective of example ordering and background knowledge presence.

Let’s take the blocks example used in incremental concept learning and solve using version spaces.

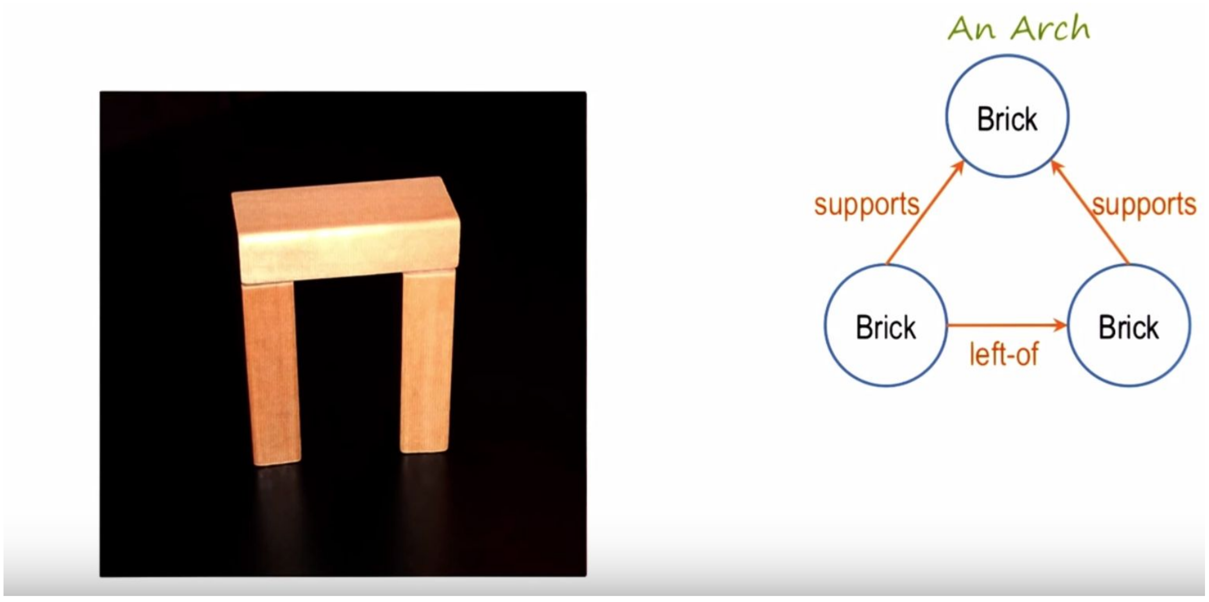


Block 1 (+ve)

Hor. count: 2
Ver. count: 2
Ver-To-Hor relation: supports
Ver. relation: left-of
Hor. separations: exists
Hor. shape: brick
Ver. shape: brick

2	[any]
2	[any]
Supports	[any]
Left-of	[any]
Exists	[any]
Brick	[any]
Brick	[any]

The most specific version on left and the most general version on right.

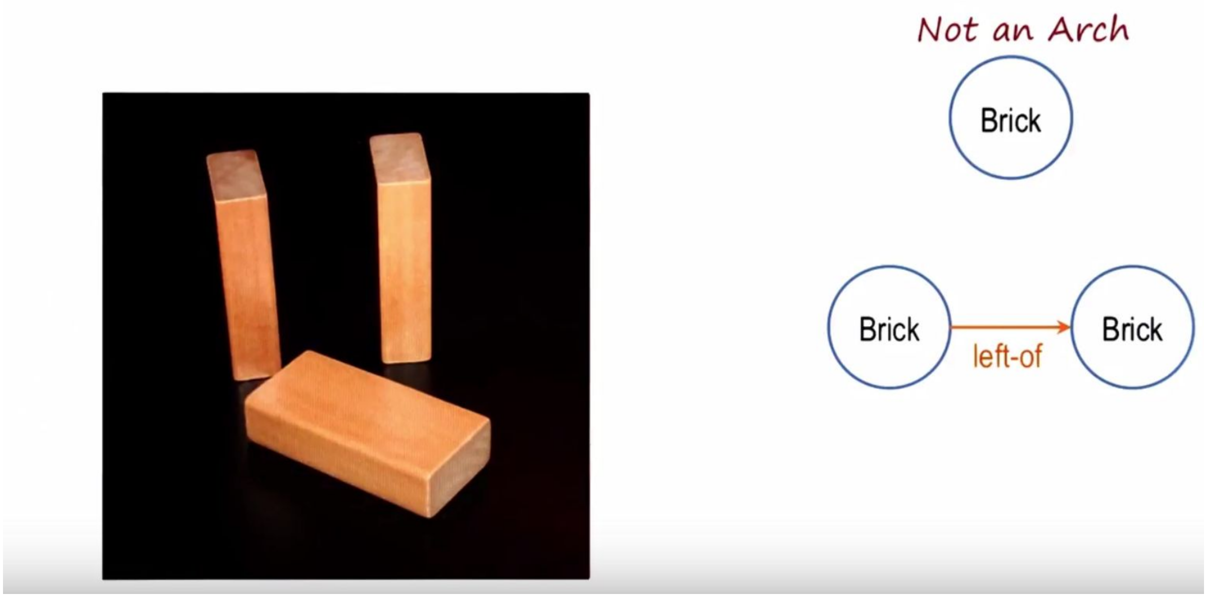


Block 2 (+ve)

Hor. count: 1
Ver. count: 2
Ver-To-Hor relation: supports
Ver. relation: left-of
Hor. separations: exists
Hor. shape: brick
Ver. shape: brick

2	[any]	[any]
2	2	[any]
Supports	Supports	[any]
Left-of	Left-of	[any]
Exists	Exists	[any]
Brick	Brick	[any]
Brick	Brick	[any]

Since, it’s a positive example, the specific version on left is generalized.



Block 3 (-ve)

Hor. count: 1
Ver. count: 2
Ver-To-Hor relation: no-support
Ver. relation: left-of
Hor. separations: exists
Hor. shape: brick
Ver. shape: brick

[2]
[any]
[any]
[any]
[any]
[any]
[any]

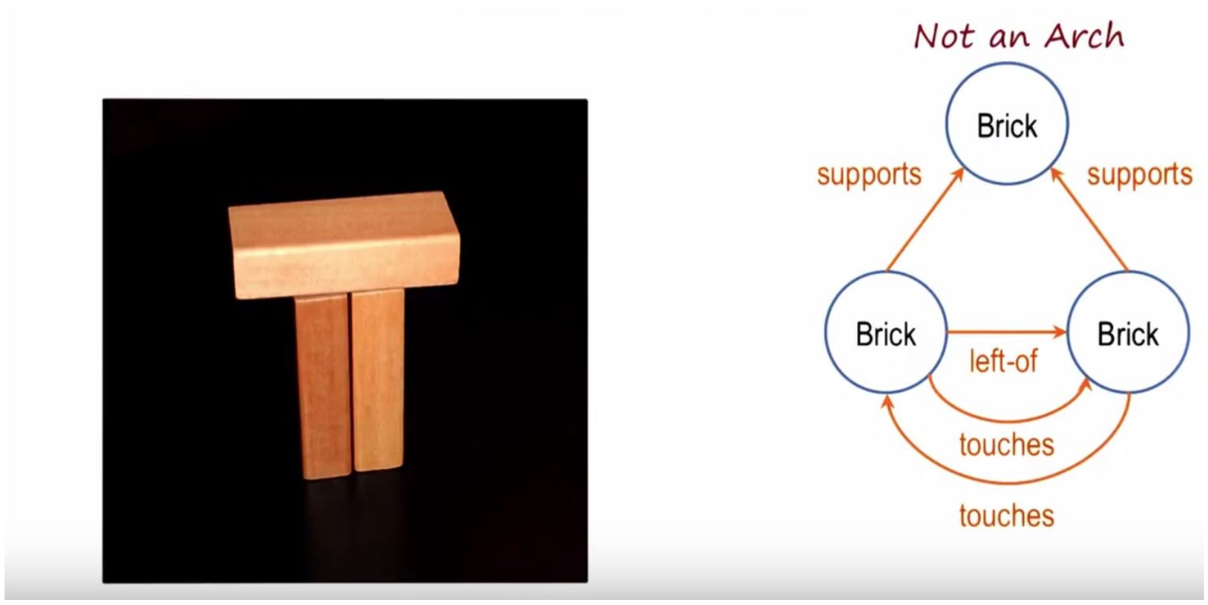
2
2
Supports
Left-of
Exists
Brick
Brick

[any]
2
Supports
Left-of
Exists
Brick
Brick

[any]
[any]
[any]
[any]
[any]
[any]
[any]

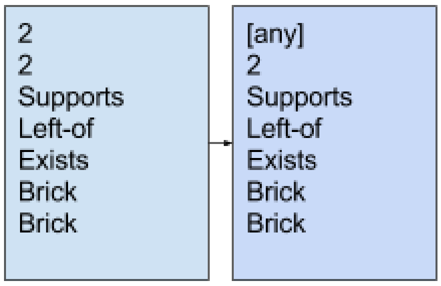
[any]
[any]
[Supports]
[any]
[any]
[any]
[any]

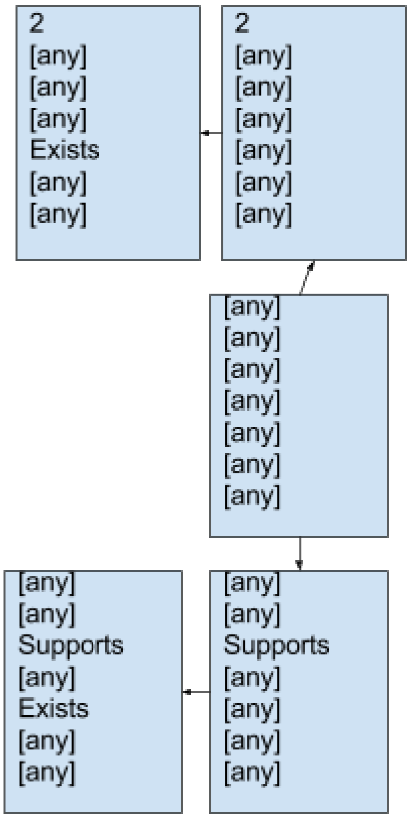
Since, it’s a negative example, the generalized version on right is specialized.



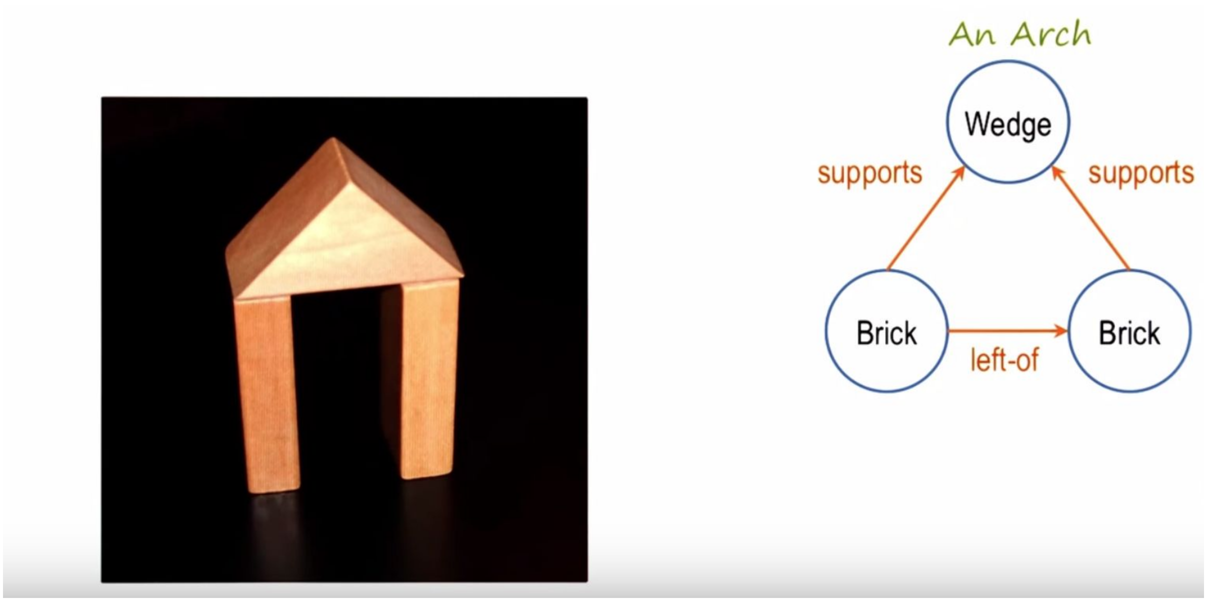
Block 4 (-ve)

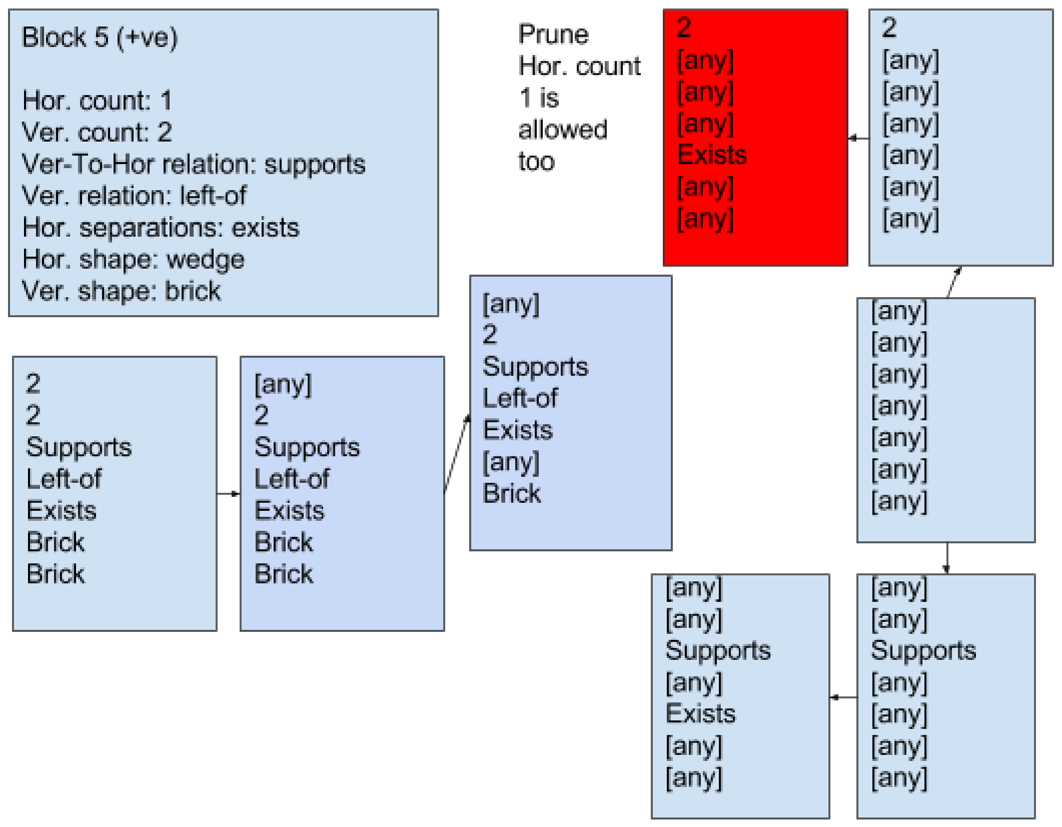
Hor. Min count: 1
Ver. Min count: 2
Ver-To-Hor relation: no-support
Ver. relation: left-of
Hor. separations: not exist
Hor. shape: brick
Ver. shape: brick





Since it's a negative example, the generalized versions on right are specialized.




Block 5 (+ve)
Hor. count: 1
Ver. count: 2
Ver-To-Hor relation: supports
Ver. relation: left-of
Hor. separations: exists
Hor. shape: wedge
Ver. shape: brick
Prune Hor. count 1 is allowed too
2 [any] [any] [any] Exists [any] [any]
2 [any] [any] [any] [any] [any] [any]
[any] 2 Supports Left-of Exists [any] Brick
[any] [any] [any] [any] [any] [any] [any]
2 2 Supports Left-of Exists Brick Brick
[any] 2 Supports Left-of Exists Brick Brick
[any] [any] Supports [any] Exists [any] [any]
[any] [any] Supports [any] [any] [any] [any]

Since it’s a positive example, the specific version on left is generalized.

So, the generalized answer converged from both direction is:

[any]
2
Supports
Left-of
Exists
[any]
Brick

Convered Generalization: Any number of horizontal blocks of any shape put on two vertical bricks placed one left of another, supporting the horizontal blocks, with a separation existing between vertical blocks.

Question 2: *We've seen various means of representing and understanding knowledge. That understanding is often based on the context in which that knowledge is learned. However, given a different context, that understanding may not be valid. Demonstrate two examples, in the terms of our KBAI concepts like frames and common sense reasoning, in which the knowledge and understanding gained in one context would lead to an invalid understanding within a different context.*

Sol:

Let's consider a sentence, "I was wondering why the ball was getting bigger, and then it hit me". Here the word 'hit' can mean that the ball physically struck me or it occurred to me why the ball was getting bigger after some thought. The knowledge representation in a frame taking 'hit' in one context would go wrong in the other context. Same with another example using the word 'take' - "It's hard to explain puns to kleptomaniacs because they take things literally". Here, the word 'take' can mean actually physically taking/stealing things or to interpret things literally. The word 'take' can take various meanings as follows:

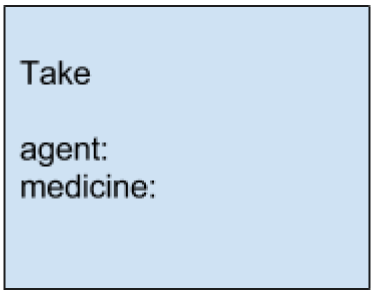
Take (verb)

- To remove
- To bring or transport
- To capture
- To medicate
- To steal
- To subtract
- To assume control
- To measure
- To cheat
- To accompany etc.

Example sentences: Doctor took my blood pressure (To measure), Ashok took Aspirin (To medicate) etc.

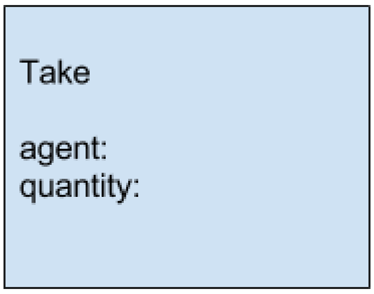
Each of the above meanings for 'take' has different knowledge representations as follows:

Take (to medicate)



Eg: Ashok took Aspirin

Take (to measure)



Eg: Doctor took my blood pressure

Based on the background knowledge of different words in a sentence, we pick the right knowledge representation of the word ‘take’. So, the representation for take meaning ‘to medicate’, goes completely wrong in a different context. For example, “Sunil took candy from the baby” has a different context. From the background knowledge of the word ‘candy’, we know it’s not a medicine and the representation we have in ‘To medicate’ sense for word ‘take’ goes wrong here.

Let’s consider following two sentences using the word ‘kill’ in different contexts.

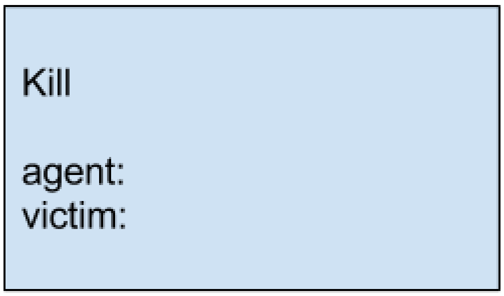
Sentence1: The Tsunami killed 45 people in India
Sentence2: The president of India killed 45 proposals for Tsunami predictions

First the background knowledge tells that the word ‘kill’ has different meanings.

Kill (verb):

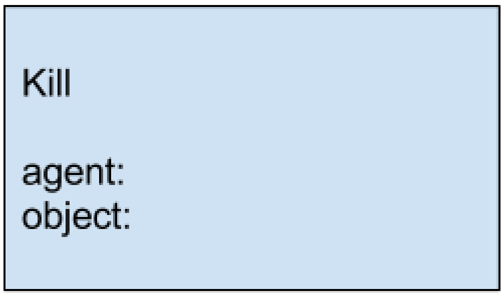
- To cause the death of
- To put an end to

Kill (to cause the death of)



Eg: Sentence 1

Kill (to put an end to)



Eg: Sentence 2

The representation we have for kill in ‘to cause the death of’ sense can completely go wrong for sentence 2 where there is usage of same word ‘kill’, but in different sense. Hence, the representations are context specific.

Question 4: *A common mistake we saw during the assignments in the middle portion of this course was the usage of scripts as a way of describing effectively any algorithm. Like algorithms, scripts consist of repeated steps or actions over input that may differ, but fits certain structures. For example, an algorithm may be able to operate over any real numbers, and a script may be able to operate within any fast food restaurant. However, treating scripts as a representation for algorithms risks losing the value and usefulness of scripts. Contrast scripts with algorithms. Comment specifically on the strengths and weaknesses of scripts, and the kinds of problems for which they’re suited.*

Sol:

Scripts allow us to make sense of discourses, stories and scenes. Theory of scripts can be built from the frames and common sense reasoning. A script is a structure that suggests a set of circumstances which could be expected to follow on from one another. It’s similar to a thought sequence or chain of situations that can be anticipated. Whereas, algorithm is just a set of steps to solve a problem.

Scripts are strengthy because:

- Causal relationships between events exist
- Entry conditions exist that allow an event to occur
- Events tend to happen in known patterns or runs
- Prerequisites exist upon events taking place e.g. when a purchaser buys a house

Different components of scripts include:

- Entry conditions: must be satisfied before events in script occur
- Results: Conditions that can be true after events in script occur
- Props: Slots representing objects involved in events
- Roles: persons involved in the events
- Tracks: variations on the script. Same script components can be shared by different tracks
- Scenes: The sequence of events that occur. Events are represented in a conceptual dependency form

Algorithms are a set of rules for solving a problem in finite number of steps. eg : Finding the greater common divisor. Algorithms resemble recipes. Recipe tells you how to accomplish a task by performing a number of steps. For example, to bake a cake, the steps are: preheat the oven, mix flour, sugar and eggs, pour into a pan, and so forth. However, “algorithm” is a technical term with more specific meaning respecting following properties:

- An algorithm is an unambiguous description that makes clear what has to be implemented without any vague statements. “Bake until done” is vague. An algorithm puts it more explicit defining what “done” actually means, like “Bake until cheese melts”
- Algorithms expects defined set of inputs
- Algorithm produces defined set of outputs
- Guaranteed to terminate and produce result
- Most algorithms return correct result. And the preconditions imposed on inputs must be met

Example of an algorithm: Find the largest number among two:

- Input: A list L of positive numbers (L must contain at least one number)
- Output: A number n which is largest of numbers in L
- Algorithm:
    - Set max to 0
    - For each x in L, compare max to x. If x is larger, set max to x
    - max is now set to the largest number in the list

Example of a script usage: Scripts are useful in describing certain situations like a bank robbery case. This might involve:

- Getting a gun
- Hold up a bank
- Escape with money

Here the props might be:

- Gun, G
- Loot, L
- Bag, B
- Car to get away, C

The roles might be:

- Robber, R
- Cashier, N
- Bank manager, M
- Policeman, P

The entry conditions might be:

- R is poor
- R is impecunious

The results might be:

- R has more money
- M is angry
- N is in a state of shock
- P is shot

There are three scenes here. 1. Obtaining gun 2. Robbing bank 3. Get away in car

The full script is described as follows:

Script Robbery	Track: Successful snatch
Props - Gun, G - Loot, L - Bag, B - Car to get away, C	Roles - Robber, R - Cashier, N - Bank manager, M - Policeman, P
Entry conditions - R is poor - R is impecunious	Results - R has more money - M is angry

	- N is in a state of shock - P is shot
Scene 1: Obtaining gun - R enters gun shop - R chooses gun - R buys gun	
Scene 2: Robbing bank - R enters bank - R has an eye on M, N, P - R moves to N’s position - R takes out gun - R says “Give me money or else I’ll..” to N - P says “Hands up” to R - R shoots with G - P ingests bullet from G - N takes out property, L - N puts L in bag, B - R leaves with L - M raises alarm	
Scene 3: - R takes car and escapes	

Conclusion: Scripts provide the ability to predict events. Scripts provide a single coherent interpretation built up from a collection of observations whereas algorithms define just a series of steps to solve a problem. However, scripts have some disadvantages compared to other representations. Scripts are less general compared to other representations like frames. Scripts may not be able to represent all kinds of knowledge.