Lesson 1: Intro to KBAI

- Conundrums in and characteristics of AI
- KBAI and the 4 schools of AI
- Cognitive Systems
- Topics in AI

Conundrums in AI

- Intelligent agents have limited computational resources. How can we get performance?
- Computation is local, but most AI problems have global constraints
- Logic is fundamentally deductive, but many problems are not.
- The world is dynamic, but knowledge is limited. How can an AI agent address a new problem?
- Problem solving, reasoning, and learning are complex, but explanation and justification are even more complex.

Characteristics of AI Problems

- Knowledge often arrives incrementally, not all at once.
- Problems exhibit recurring patterns.
- Problems have multiple levels of granularity.
- Many problems are computationally intractable.
- The world is dynamic, but knowledge of the world is static.
- World is open-ended but knowledge is limited.

Characteristics of AI Agents

- Agents have limited computing power.
- Agents have limited sensors.
- Agents have limited attention. Can't focus on everything at once.
- Computational logic is fundamentally deductive.
- AI agents' knowledge is incomplete relative to the world.

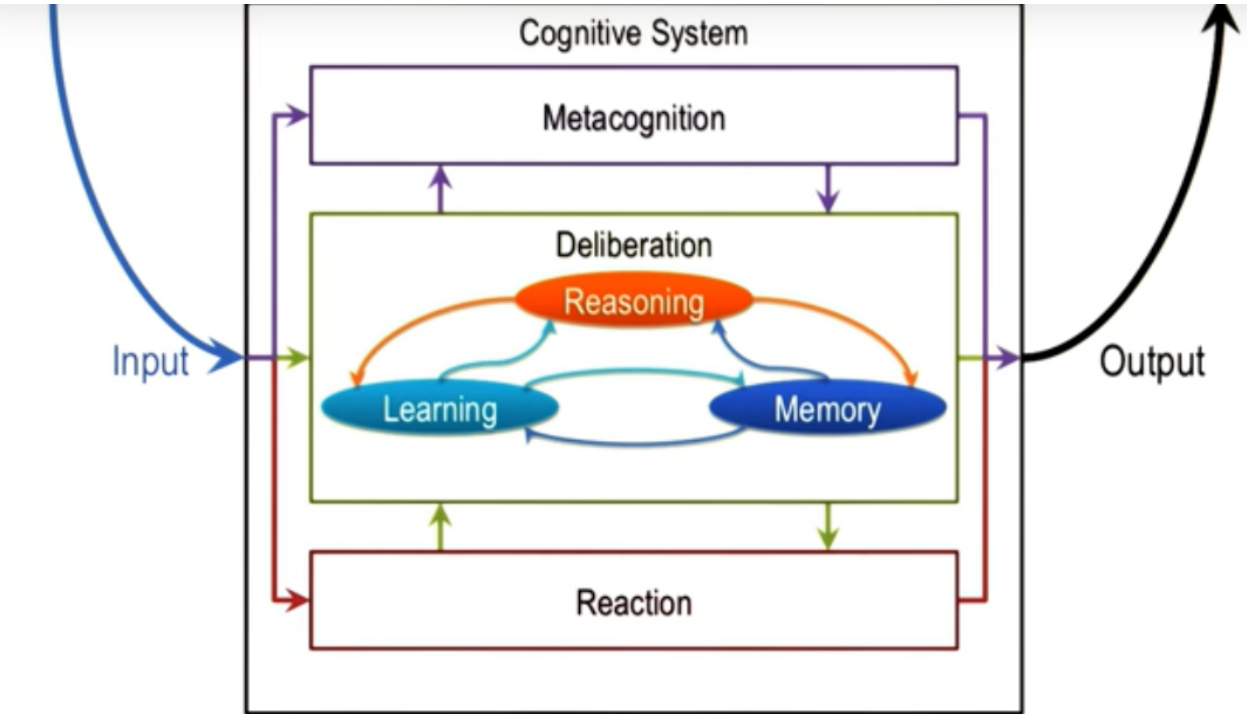
How does Watson play Jeopardy?

- Read the clue
- Search through knowledge base
- Decide answer
- Phrase answer in form of a question

What is KBAI?

Reasoning: understand natural language, generate national language, make decisions
Learning: gets the right answer sometimes, stores it, makes note of mistakes
Memory: If you're going to learn, resulting knowledge has to be stored somewhere. If you're going to reason using knowledge, it has to be accessed from somewhere.

As a group, called "Deliberation Process". One part of the overall architecture of an AI agent.



Four Schools of AI

Agents that think optimally (machine learning, Google Maps)
Agents that act optimally (airplane autopilot, Roomba)
Agents that think like humans (semantic web, Siri)
Agents that act like humans (improvisational robots, C-3PO)

Course materials: AI: A modern Approach Ch. 1 Section 1.

What are Cognitive Systems?

- Cognitive: dealing with human-like intelligents
- Systems: multiple interactive components like learning, reasoning, memory
- Cognitive systems: systems that exhibit human-like intelligence through processes like learning, reasoning, and memory.

Cognitive System Architecture

World -> Input -> Cognitive System -> Output -> World

So far we have talked about a single cognitive system, but there can be multiple systems that interact with one another.

Inside of a cognitive system, takes input information about the world, returns output as actions on the world.

Reaction (no planning, direct mapping of input to action, e.g. slamming on brakes once lights of car in front of you turn red)

Deliberation (come up with a plan to determine action to take, using reasoning, learning, and memory, e.g. determining whether to change lanes)

Metacognition (reasoning about the internal mental world, reasoning about reaction or deliberation, e.g. changing lanes at the wrong time causing cars to honk - may reconsider deliberation)

Topics in KBAI (8 units):

Fundamentals: Semantic Networks, Generate and Test, Production Systems, Means-End Analysis, Problem Reduction

Planning: Logic as a Knowledge Representation & Systematic Planning

Common Sense Reasoning: Frames, Understanding, Common Sense Reasoning, Scripts

Analogical Reasoning: Learning By Recording Cases, Case-Based Reasoning, Explanation-Based Learning, Analogical Reasoning

Metacognition: Learning by Correcting Mistakes, Meta-Reasoning, Ethics in AI

Design and Creativity: Configuration, Diagnosis, Design, Creativity

Visuospatial Reasoning: Constraint Propagation, Visuospatial Reasoning

Learning: Learning by Recording Cases, Incremental Concept Learning, Classification, Version Spaces

Lesson 2: Intro to CS7637

Class Goals

- Core methods of KBAI
- Tasks addressed by KBAI
- How KBAI use these methods to address those tasks
- Relationship between AI and human cognition

Learning Outcomes

- Design and implement a knowledge-based AI agent
- Use these strategies to address complex, practical problems
- Use design to reflect on human cognition

Raven's Progressive Matrices and Project

Principles of KBAI

- KBAI Agents represent and organize knowledge into structures to guide and support reasoning
- Learning in KBAI agents is often incremental
- Reasoning is top-down as well as bottom-up
- KBAI agents match methods to tasks
- KBAI agents use heuristics to find solutions that are good enough, not necessarily optimal
- Agents make use of recurring patterns in the problems they solve
- The architecture of KBAI enables reasoning, learning, and memory to support and constrain each other.

Sources:

Winston, P. (1993). *Artificial Intelligence* (3rd ed.). Addision-Wesley.

Stefik, M. (1995). *Knowledge Systems.* Morgan Kauffman: San Fransisco.

Rich, E., & Knight, K. (1991). *Artificial intelligence*. McGraw-Hill, New York.

Russell, S. & Norvig, P. (1995). *Artificial Intelligence: A modern approach.* Prentice-Hall: Englewood Cliffs.

Lesson 3: Semantic Networks

- Knowledge Representations

- Semantic Networks
- Problem solving with semantic networks
- Represent & Reason

Representations

What is a knowledge representation?
Knowledge representations have:

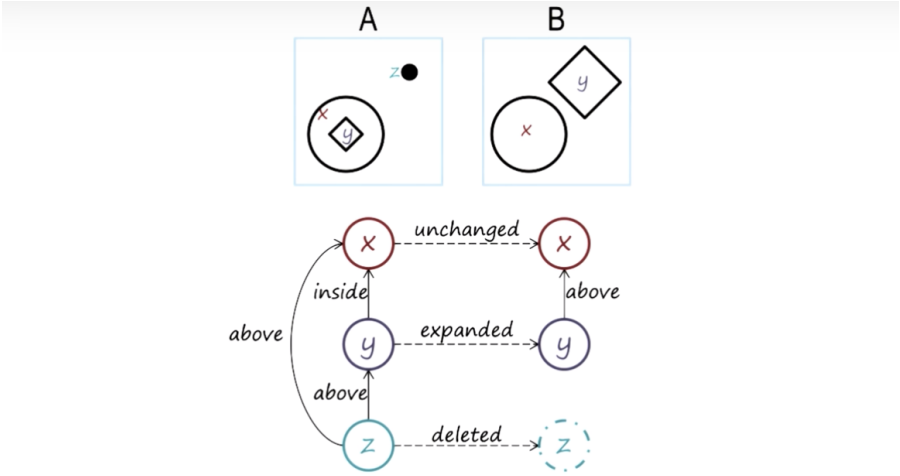
- Language (with vocabulary)
- Content (knowledge, expressed in that language)

For example, Newton's second law of motion is described as: f = ma
This representation has a language (an algebraic equation) and content (force = mass times acceleration)

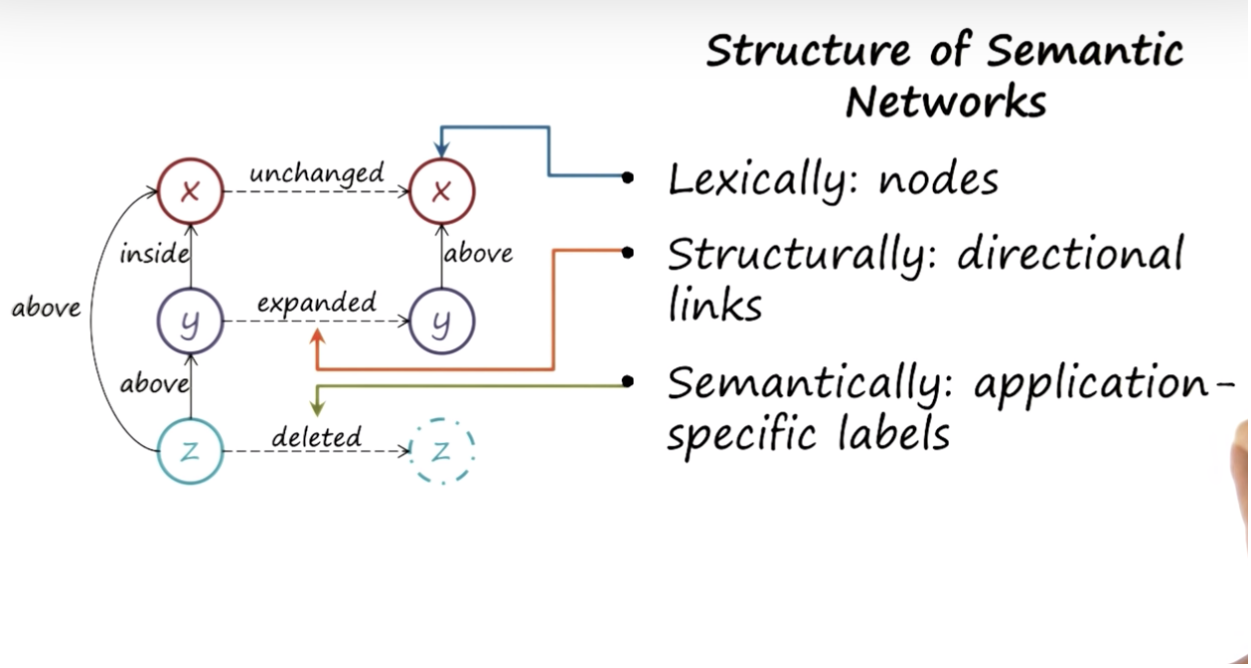
Semantic Networks

Example: 2 x 1 matrix

Represent objects and relationships as such:



Structure of Semantic Networks
Lexically: nodes
Structurally: directional links
Semantically: application of specific labels



What makes a semantic network a “good” representation?

- Makes relationships explicit
- Exposes natural constraints
- Brings objects and relations together
- Excludes extraneous details
- Transparent, concise, complete, fast, computable

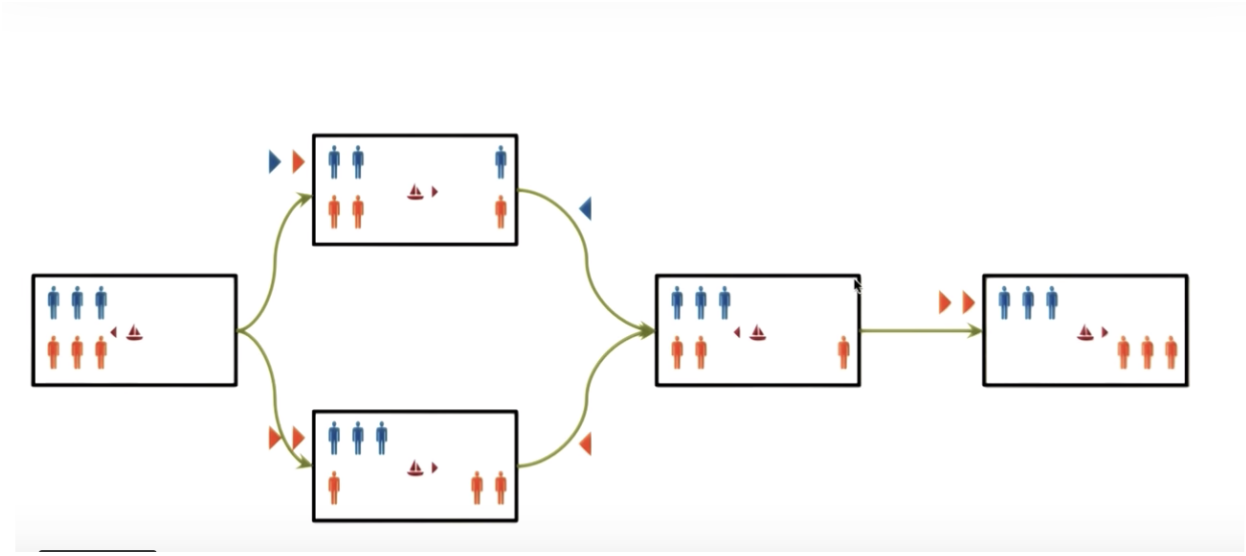
Guards and Prisoners Problem

- Also known by other names
- Originally appeared in 1200 year old textbook
- Used by many throughout AI for problem representation.

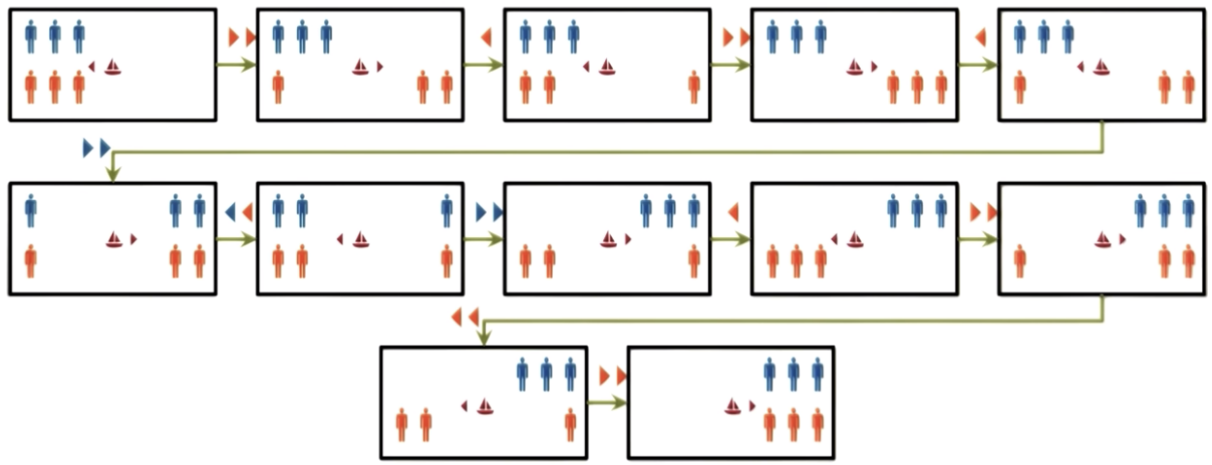
Rules:

- 3 Guards and 3 Prisoners must cross river
- Boat may only take one or two people at a time.
- Prisoners may never outnumber guards on either coast, though they may be alone on either coast.

Semantic Network for Guards and Prisoners solved together (removing invalid or redundant states):

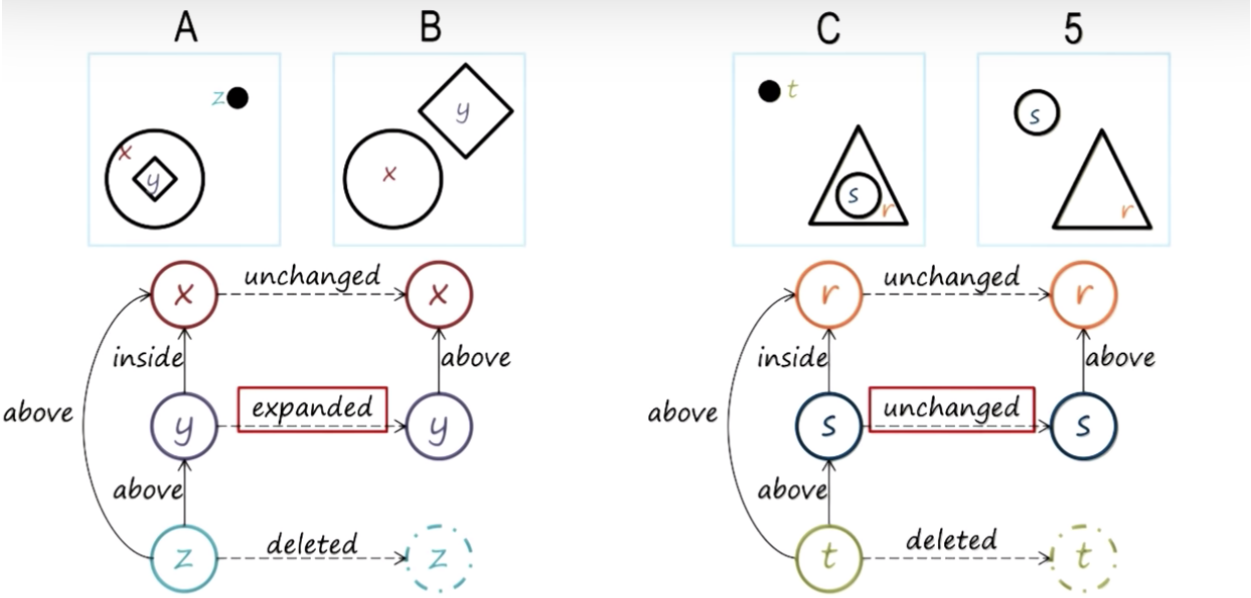


Final answer (11 moves):

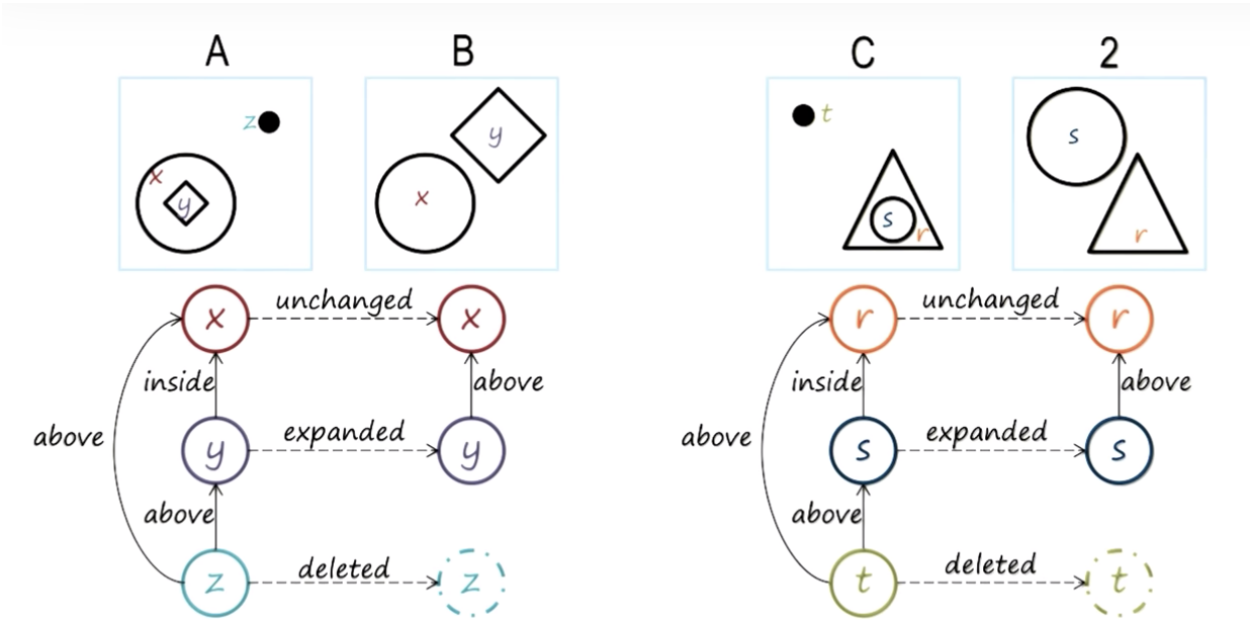


Represent and Reason

How can we use a semantic network to solve our 2x1 matrix problem?
Let's take an example response (5) - note the network does not match (y is expanded and s is unchanged...perhaps there's a better answer where we find an exact match!):



On the other hand, for (2), the network matches exactly:

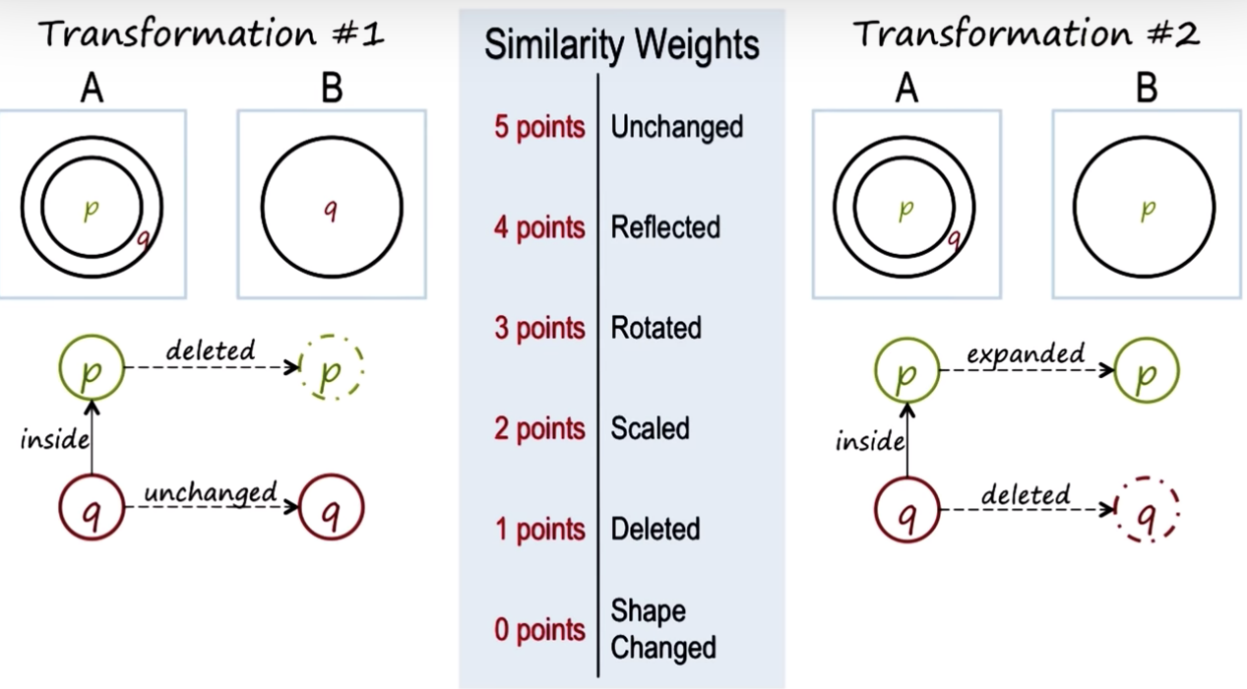


What if when building semantic networks for a problem like this, we cannot find an exact match for any answer? How could we choose? How can we evaluate how similar the matches are, or how likely they are to be correct?

Choosing matches by weights:

We can represent some transformations in multiple ways using semantic networks, for example:

How would an agent “make the call” as to which one is correct? We could generate a metric based on the “ease” of transforming one object to another, with higher numbers being “easier”.



Given this scale, what are the similarity weights?

Transformation 1: delete(p) + unchanged(q) = 1 + 5 = 6
Transformation 2: expand(p) + delete(q) = 2 + 1 = 3

So transformation 1 is “more similar” based on this metric.

Now we can see why 3 and 5 are both correct, and also why an AI agent may prefer 5 over 3.

Connections

Memory: Integral part of cognitive system.
We can say the relationship between A and B is stored in memory. Then C->1, C->2, etc. are probes into the memory, so the question becomes which one is most similar?
Reasoning: when we’re talking about transformation from A to B, and then C to a choice, one question that arose was should we make the connection between the outer or inner circle?
Correspondence problem (analogical reasoning)
Cognition: Instead of just talking about properties of objects, we’re focusing on relationships. (outside vs. inside). Focus is always on relationships and not just on objects and their features in KBAI!

Semantic Networks: Winston Chapter 2, pp. 16-32 Can be found at http://courses.csail.mit.edu/6.034f/ai3/rest.pdf

Lesson 4: Generate and Test

- Generate and test method
- Smart generators
- Smart testers
- Generate and test for Raven's Progressive Matrices and Prisoners and Guards.

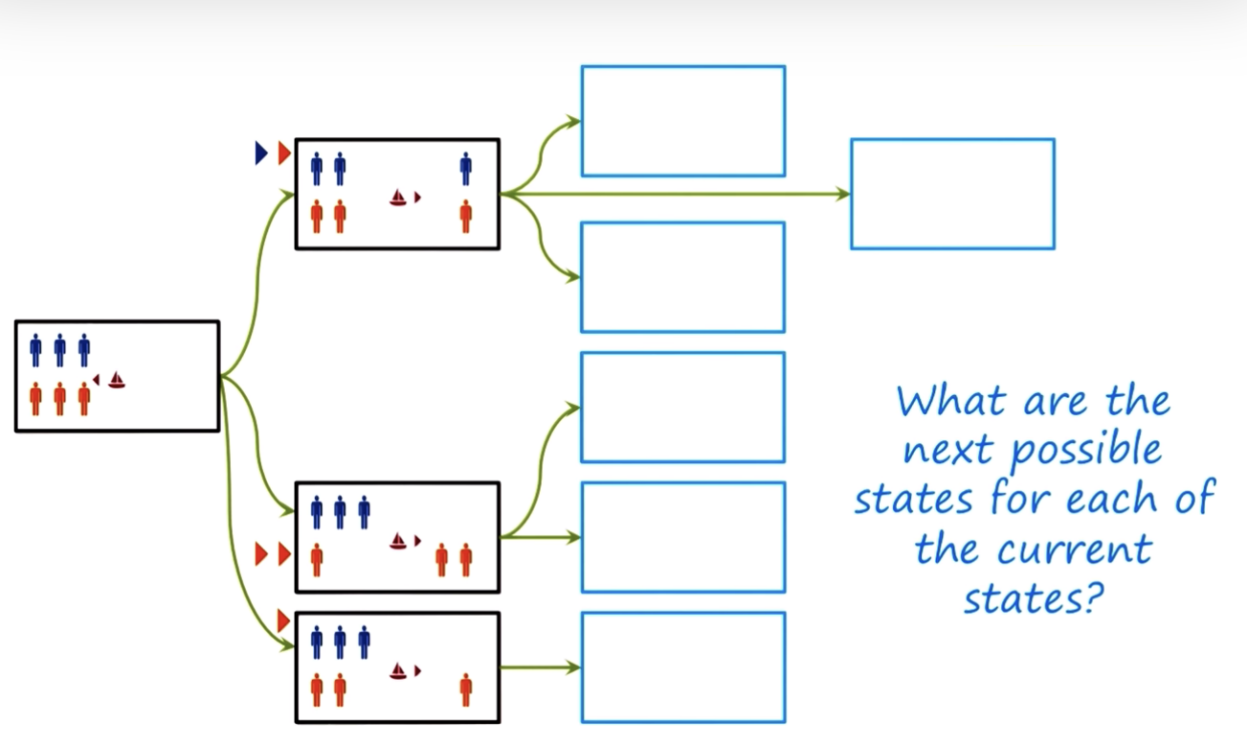
KBAI consists of 3 things:

- Knowledge representations
- Problem solving techniques/methods
- Architectures

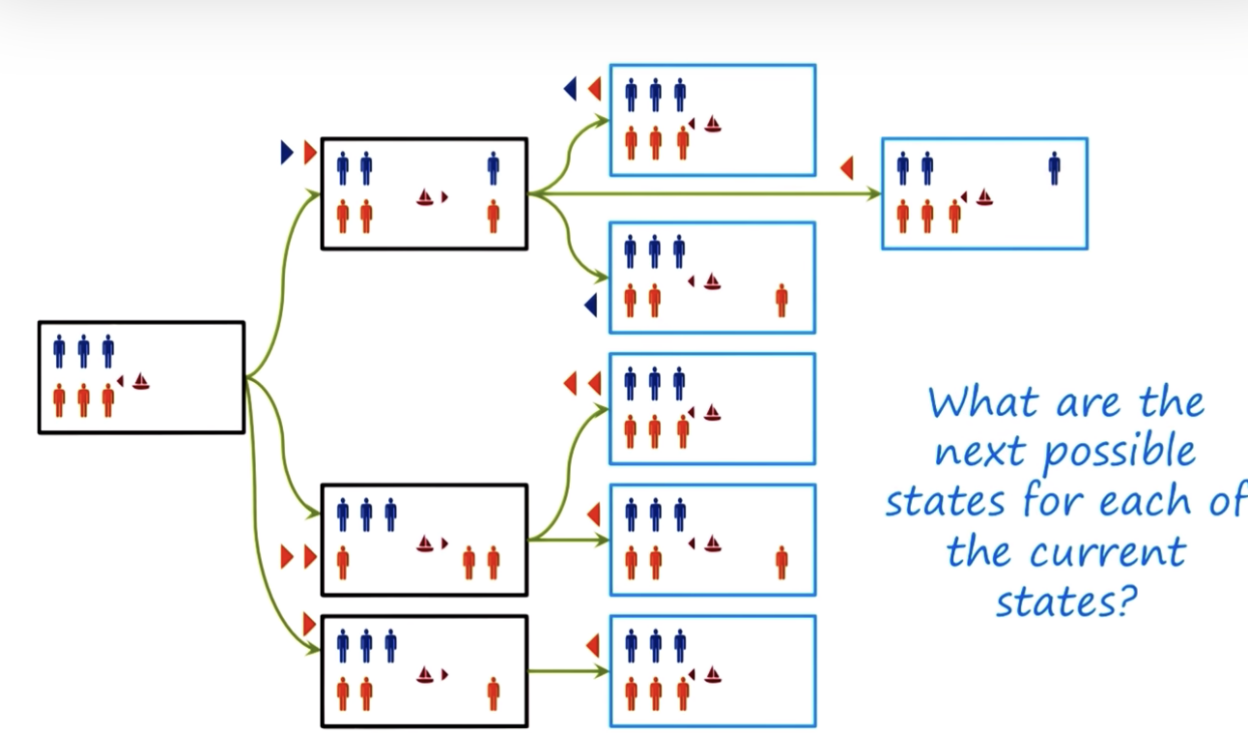
How can an AI agent generate possible successive states from a given state?
A generator can take an initial state and generate all possible successive states.
A tester removes invalid states.

"Dumb" generator generates ALL possible states. "Dumb" tester only removes states based on rules of the problem (not revisits).

So given the following network, what states would the "Dumb" generator generate for next states?



Answer:



Note that ALL possible states (including illegal and previously visited states) have been generated by the “Dumb” generator.

What states would the “Dumb” tester remove? Only those that break the rules. This is a problem and computationally VERY inefficient. We have to be able to remove states that have been visited before.

Smart Testers

Smart tester can detect whether a state is identical to a previously visited state and remove them in addition to those that break the rules.

Balance of responsibility can shift between generator and tester as well.

Smart Generators

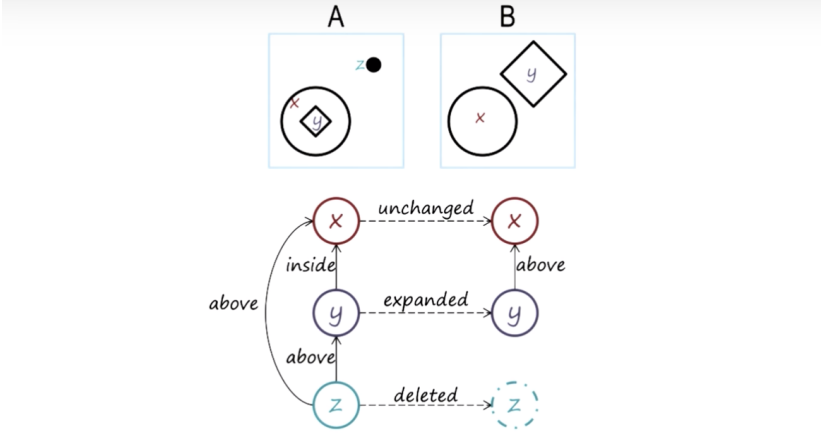
Generator can be made “smarter” to not generate states identical to the initial state. Generator could even be tweaked to not generate illegal moves.
Balance of responsibility between generator and tester can vary.

Generate and Test for Raven’s Progressive Matrices

More complicated than Guards and Prisoners, because in that problem each transformation was discrete. In the case of Raven’s Progessive Matrices, multiple changes happen between each figure. Space of possibility here is VERY large, so the need for smart generator and tester is critical.

Semantic Network for Generate and Test

Level of abstraction is important here. For example, y was expanded in this relationship. It doesn’t matter how large the shape has become. It only matters that it expanded at all. Ignores lower levels of detail.



It's not the knowledge representation that solves the problem, nor the method of solving, it's the combination of both that produces the solution.

Use the transformation between A and B, transfer that transformation to C, and use that transformation to generate a possible answer. Then we can compare with the answer choices to determine which one is most likely. See which of the choices is closest to the generated answer. In order to be the correct answer it has to meet the generated answer with a certain level of confidence. If it doesn't, we regenerate the transformation between A and B and try again.

Alternatively, could take each answer, place into D, generate the relationship between C and D, and then test it against the transformation from A to B. Find out which one of those transformations is closest.

Generate & Test: Winston Chapter 3, pp. 47-50 can be found at:http://courses.csail.mit.edu/6.034f/ai3/rest.pdf

Generate and Test with Human Cognition

Humans use it all the time.
Don't have full picture of problem nor infinite cognitive resources.
When you do not have these things, you generate solutions to a problem and test them.

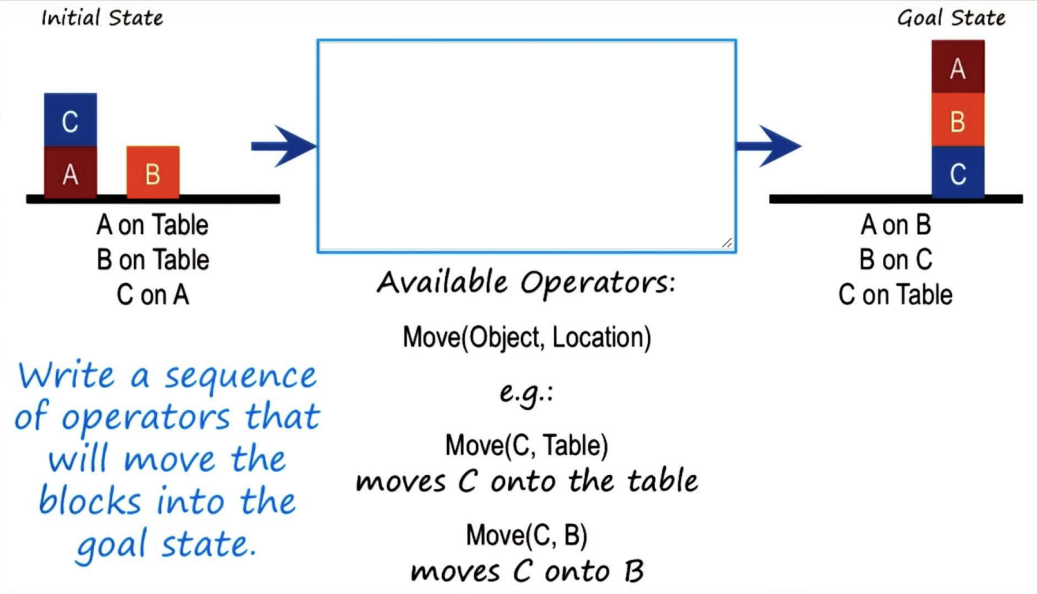
Lesson 5: Means Ends Analysis

Two other general methods of problem solving beyond Generate and Test.
These two methods are really useful for well-formed problems. Not all problems are well-formed.

These 3 methods (Generate & Test, Means-Ends Analysis, and Problem Reduction) along with semantic networks form the basic unit of all fundamental topics in the course.

- State spaces
- Means-ends analysis
- Problem solving with means-ends analysis
- Problem Reduction

Block problem:

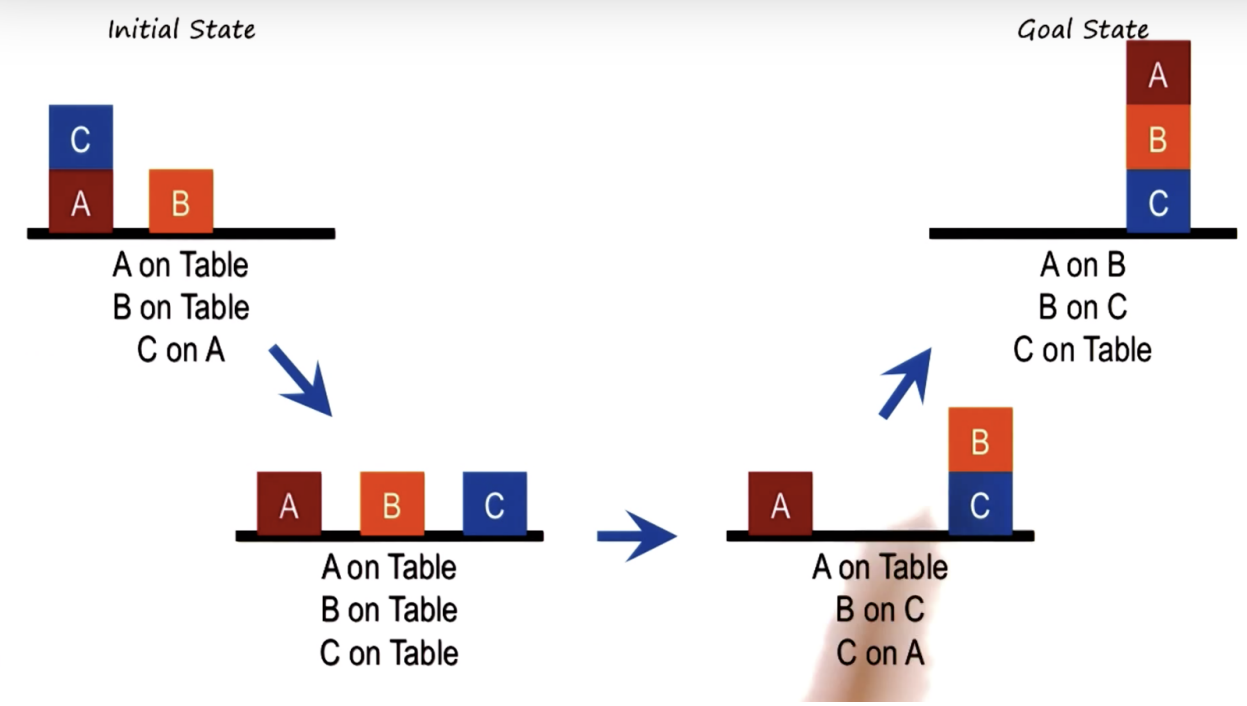


Move C to table, then B on C, then A on B.

How can you make an AI agent to do the same thing?
How does means-ends analysis come up with a particular sequence of operations?

State Spaces

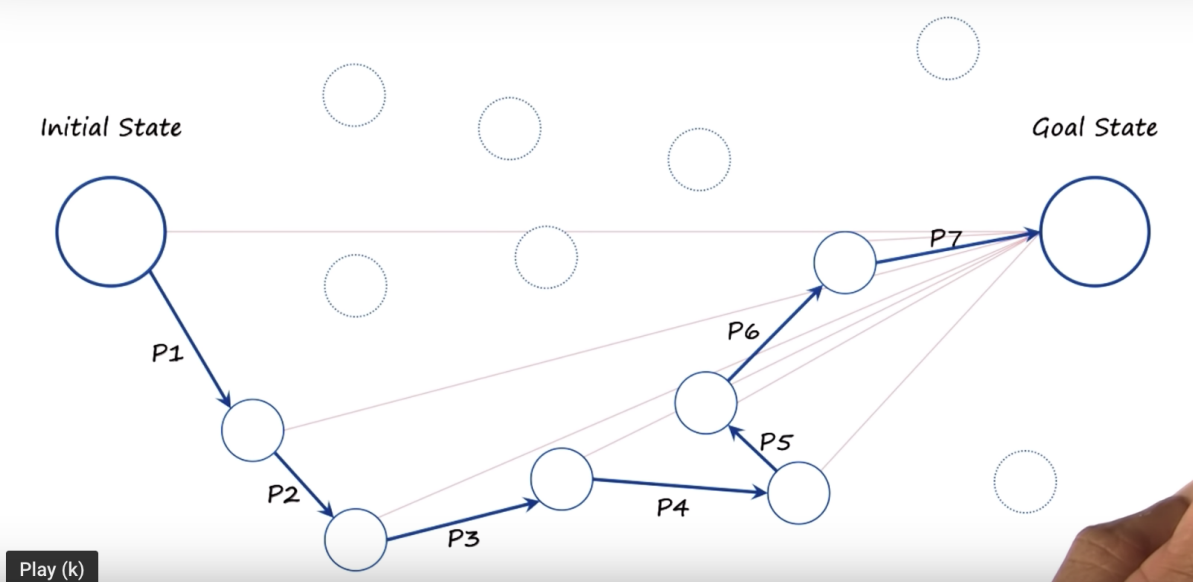
We can imagine problem solving occurring in a state space.
Initial state, goal state, and a lot of intermediate states.
How do you find a path from the initial state to the goal state?
How can an AI agent derive a path from the initial state to the goal state?



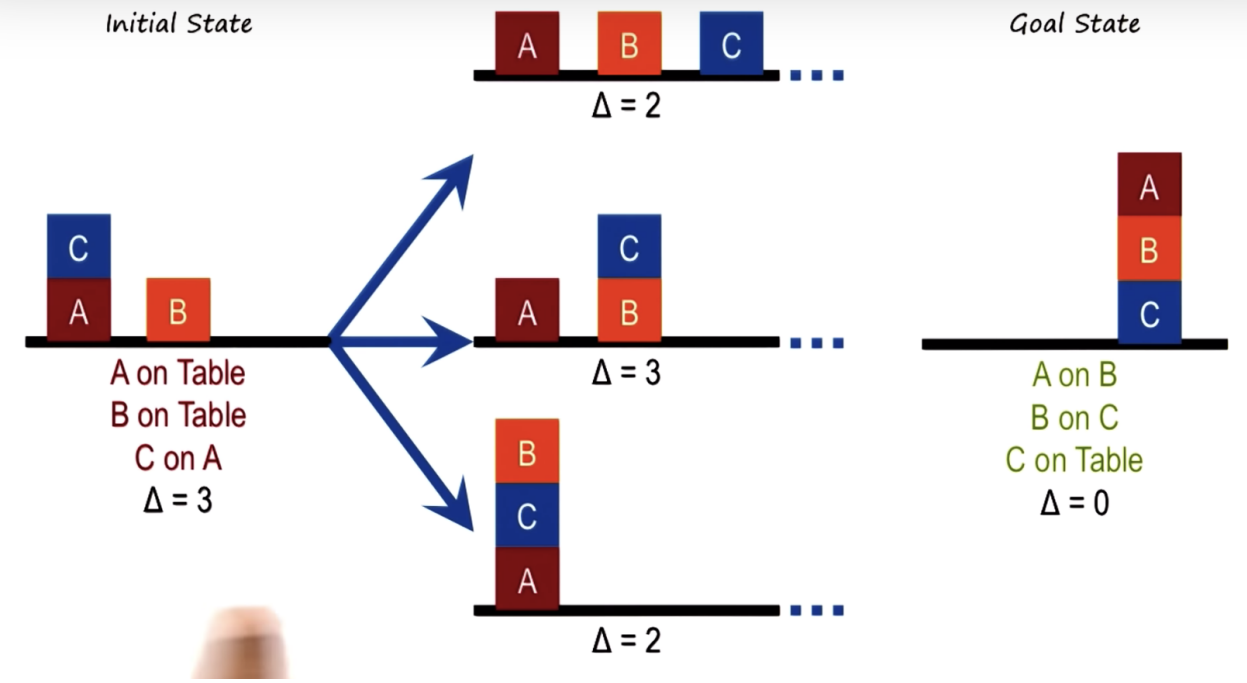
How does an AI method know which operations to select in order to get to the goal state? How does it decide at an intermediate state?

Differences in State Spaces

One way is to talk in terms of differences.
We should pick an operator that reduces the difference between current and goal state.
Reduction between the difference between current and goal is the “ends”
Application of operator is the “means”.
At any given state, I pick to transition to a state that reduces the difference between the current state and the goal state.

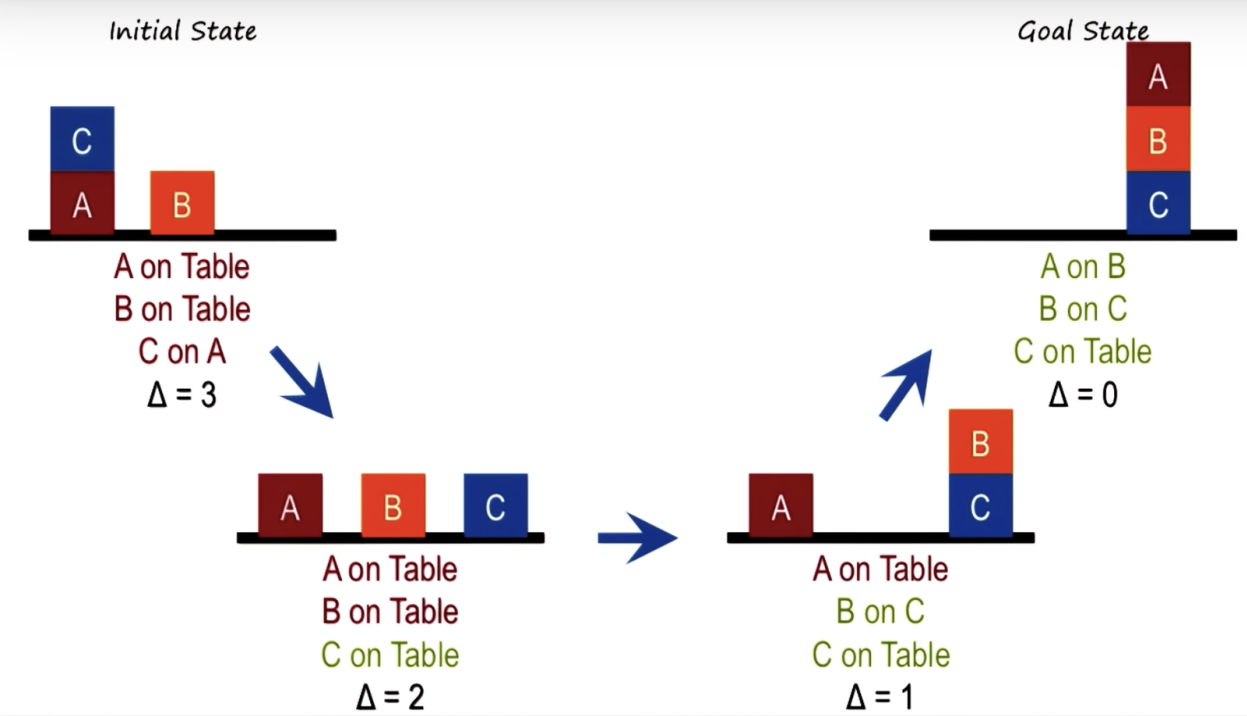


In a way, similar to pathfinding in robotics.
So in an example:



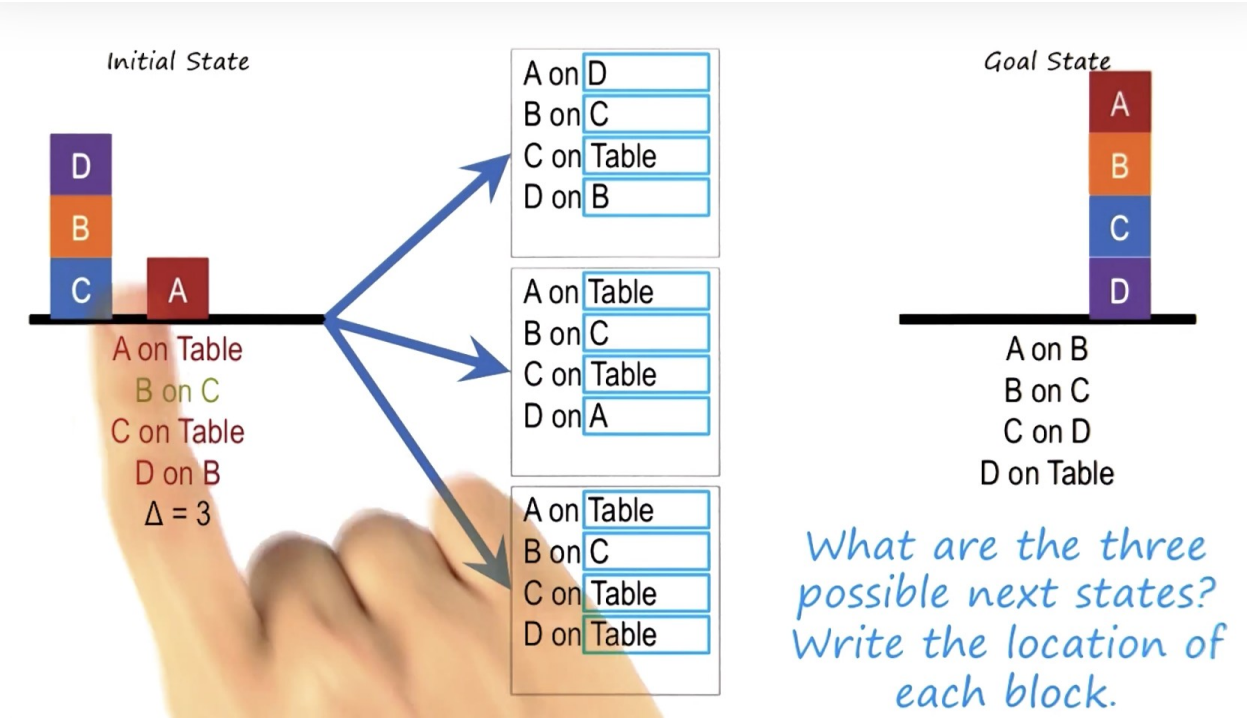
We can eliminate the middle state, but now there are two possible choices.
Means-ends analysis by itself doesn't help an AI agent break ties. This will be discussed later.

So applying this method, we can see that the difference between the current state and goal state is reduced.

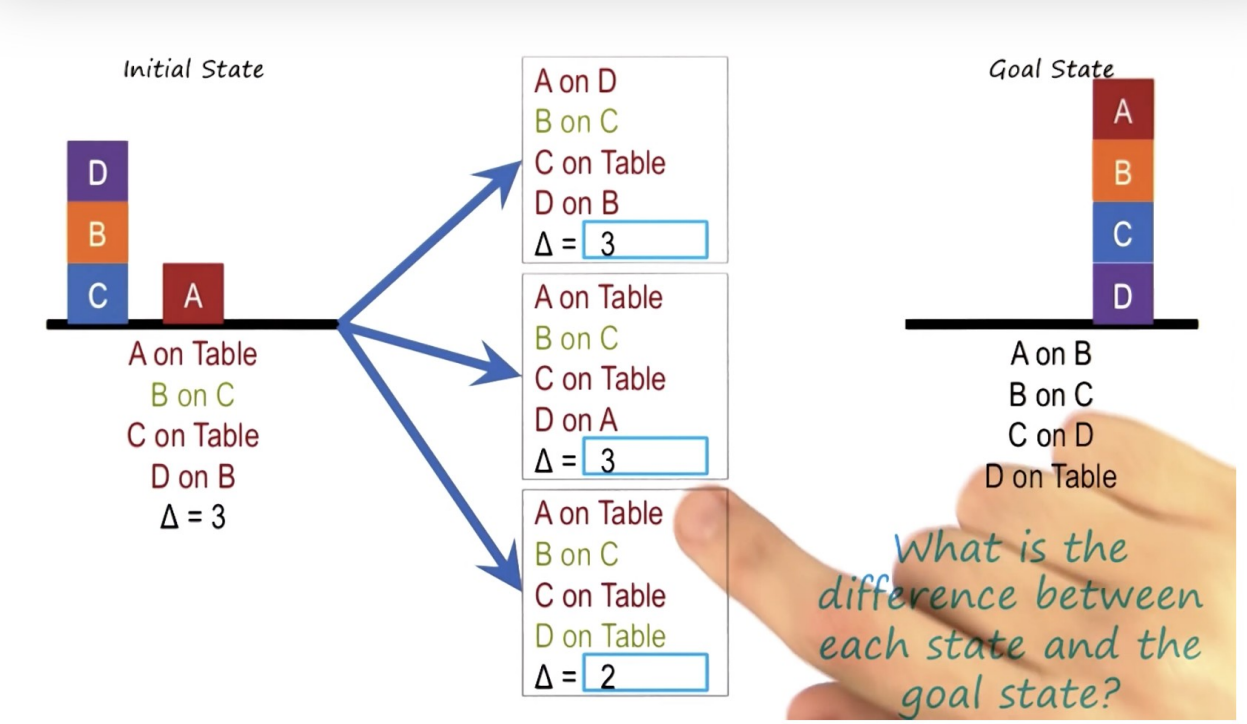


Processes of Means Ends Analysis

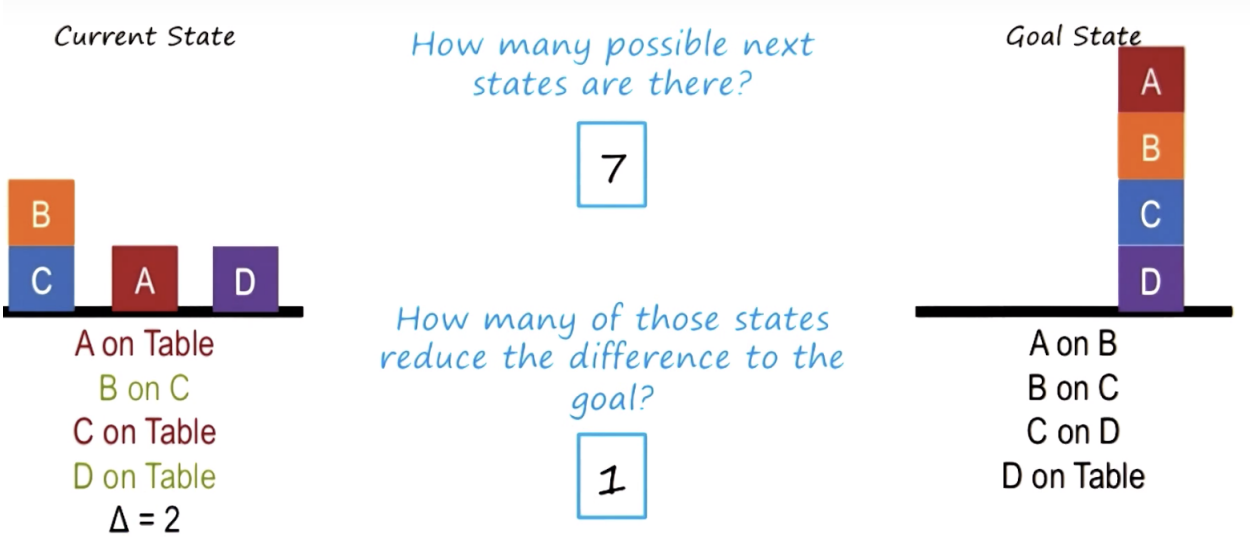
- For each operator that can be applied:
  - Apply the operator to the current state
  - Calculate difference between new state and goal state
- Prefer state that minimizes distance between new state and goal state



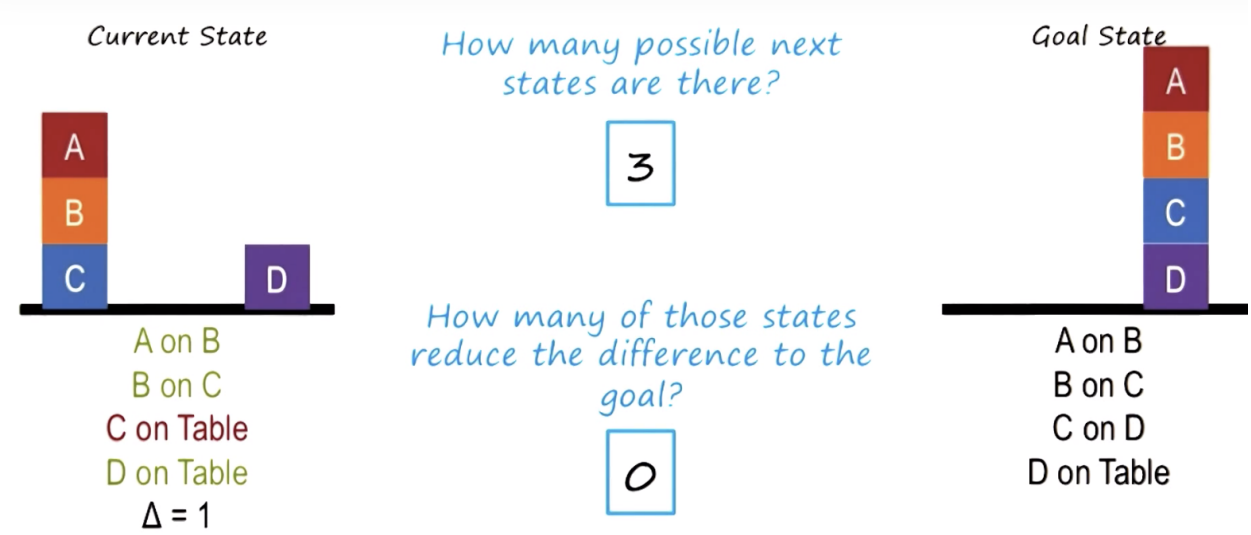
Then calculate distances between current state and goal state.



The bottom state reduces the distance from the goal state down to 2. So that state would be chosen.



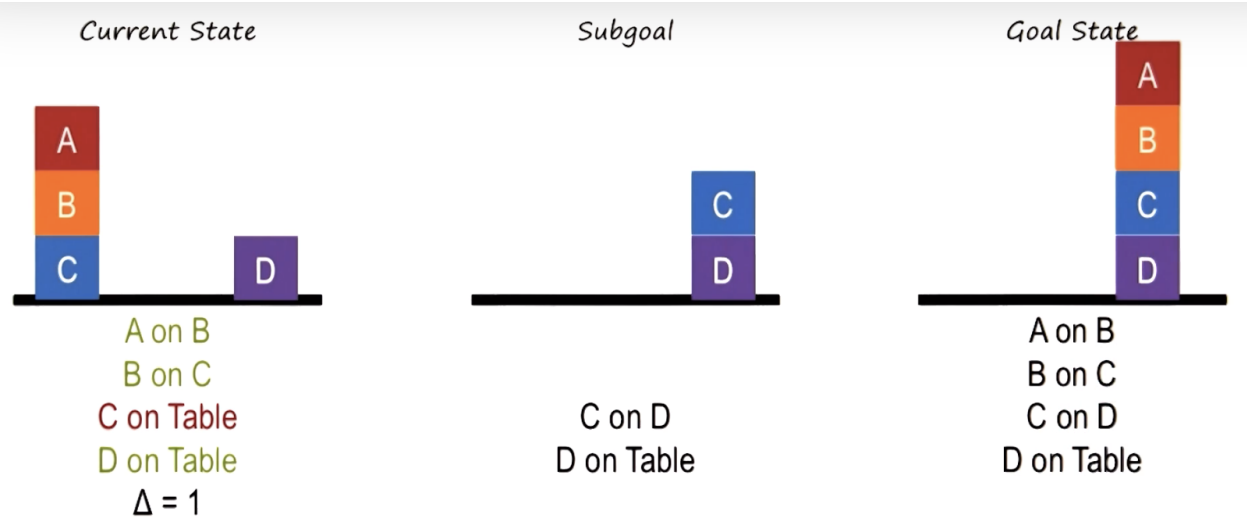
Next expand further:
But none of those states reduce distance to goal, and in fact all are a step backwards! Means-ends analysis doesn't always take us towards the goal. It can move us away from the goal, or get stuck in loops!



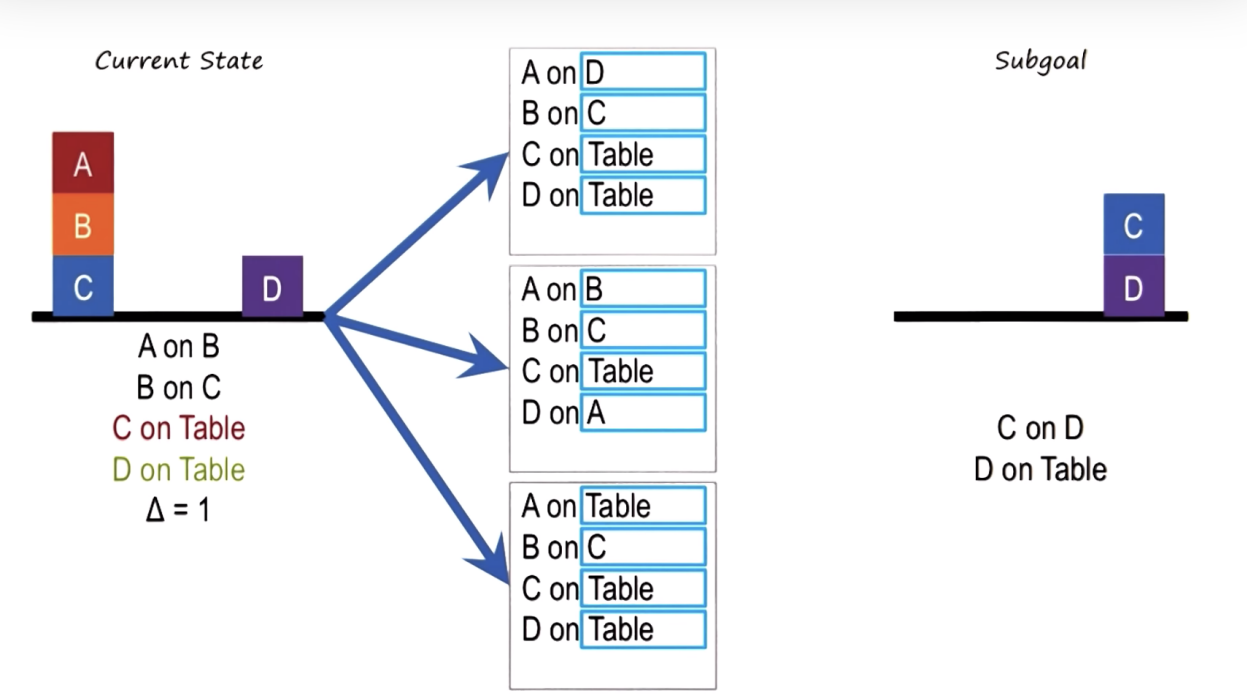
Universal AI methods like Generate and Test and Means-Ends analysis do not guarantee success and are also costly in terms of computational efficiency or optimum solutions. They do apply to a much larger class of problems, however.

Problem Reduction

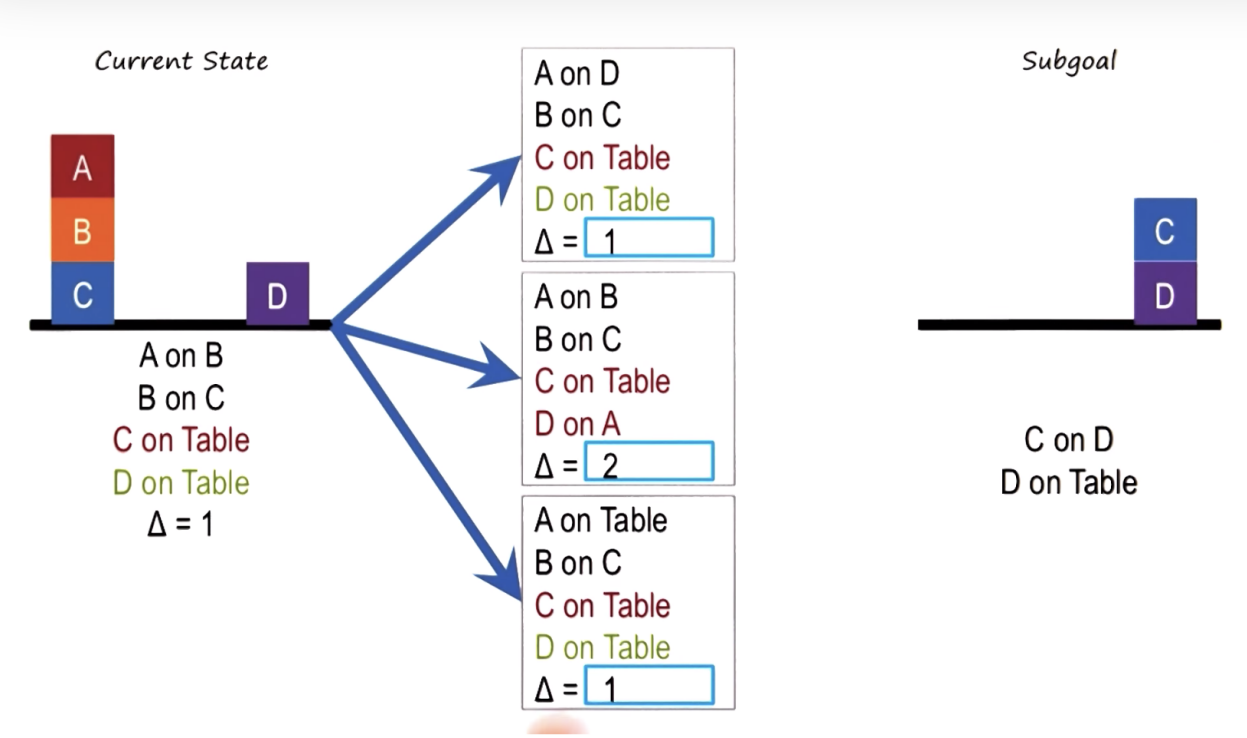
Given a hard, complex problem, reduce it into smaller problems that are easier to solve.
How do you do this? How do you compose the sub-solutions into the problem as a whole?
Let's apply this method to the block problem in the last step of means-ends analysis:
Break the goal state into its subgoals, then try to transform the current state and attack each subgoal one at a time.



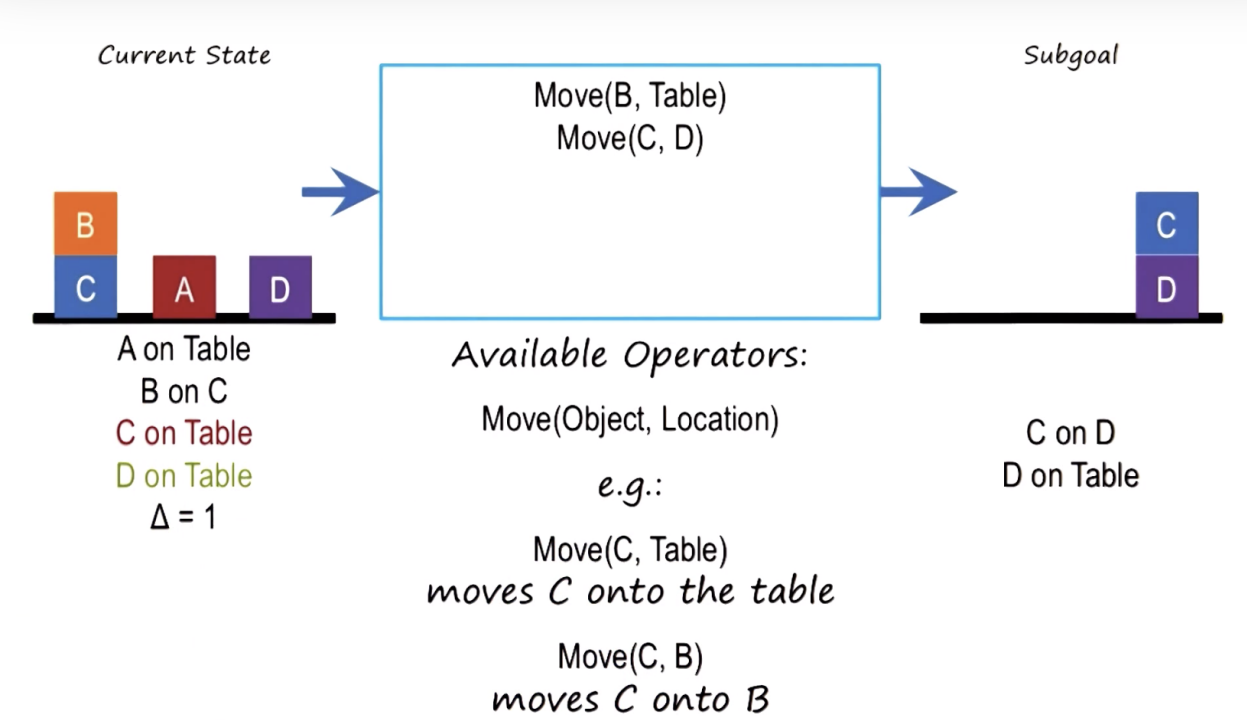
Apply means-ends analysis:



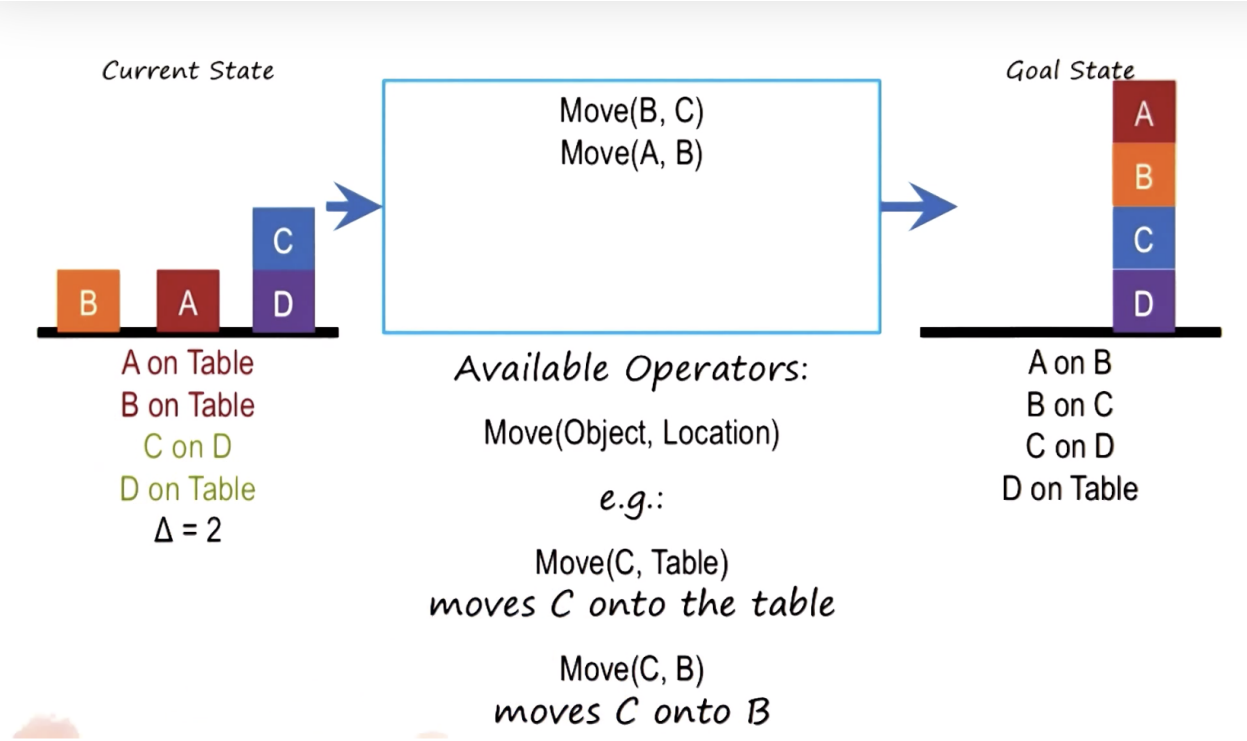
Calculate distances:



We can choose the bottom-most state (course will explain later why this is the case jin terms of breaking the tie) and expand from there.

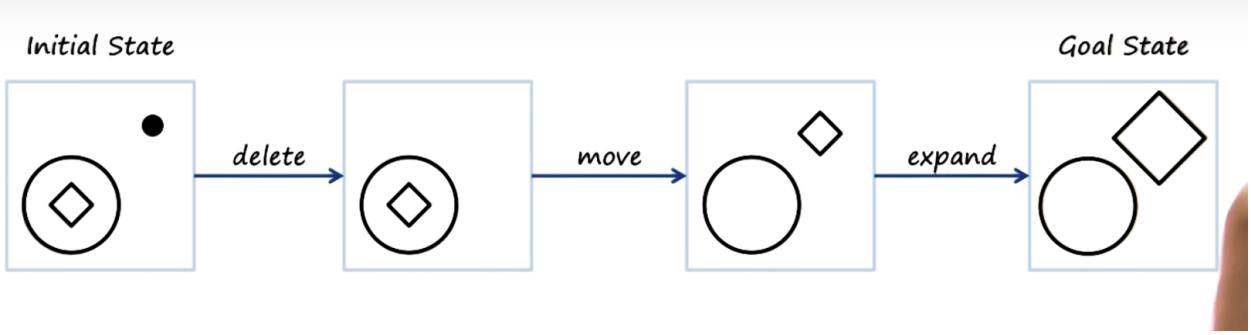


Then we can make two more moves to complete the problem (by using the other subgoals)..

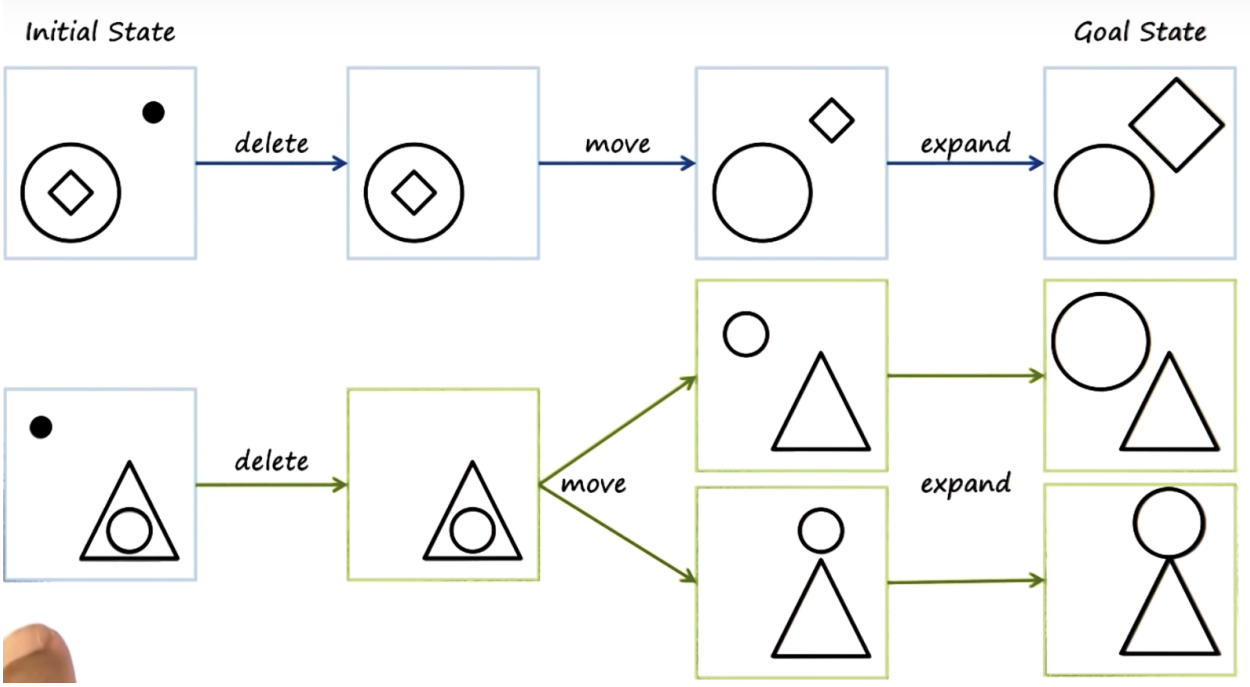


Means Ends Analysis for RPM

Each operator is a transformation. Sequence of transformations will result in full transform into answer.



What in image A corresponds to image C...now it gets a little trickier?



Also using Generate and Test and Problem Reduction (3 subgoals: transform A to B, apply transformations to C, compare against choices given to us)

Means-Ends Analysis, Problem Reduction, Generate and Test, etc. are called “weak” methods, they use very little knowledge about the world when solving problems. Later, “strong” methods will be introduced that utilize more knowledge about the world to come up with efficient/optimal solutions. Those require knowledge, however, which is not always available.

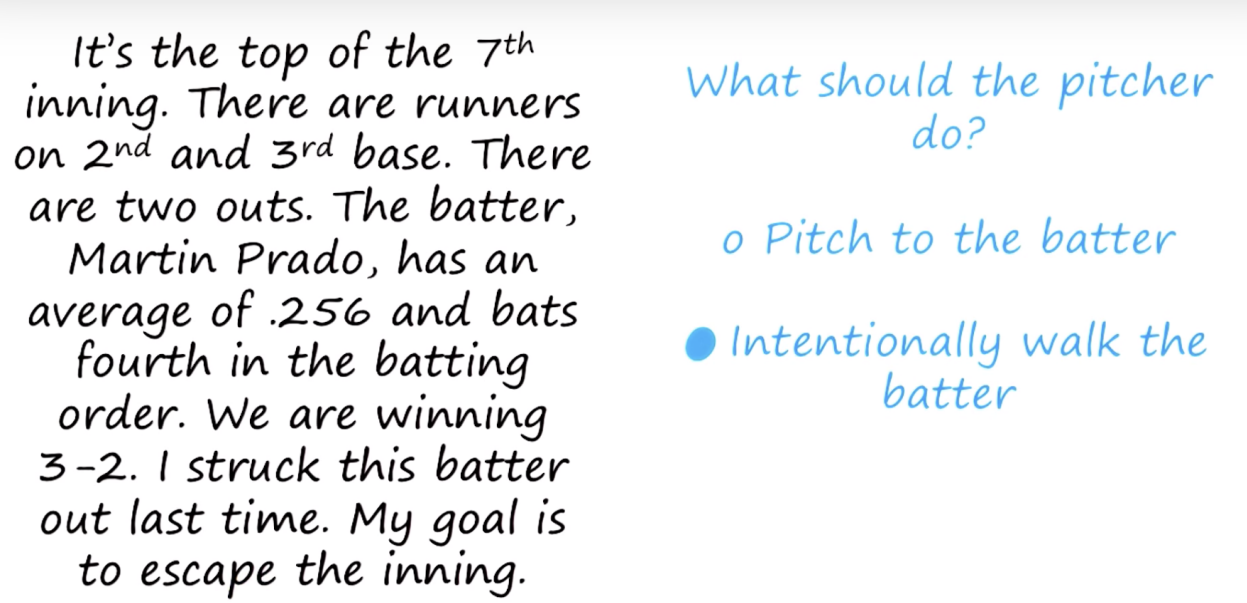
When humans work in an area they are experts in, they use knowledge intensive methods to solve problems. For areas where they have less knowledge, they rely on less knowledge intensive methods.

Lesson 6: Production Systems

Cognitive architecture in which knowledge is represented in terms of rules.

- Cognitive architectures
- Production systems
- Chunking

Baseball exercise: what should the pitcher do?



Intentionally walk the batter due to force out potential when bases are loaded. David had a lot of knowledge about baseball and used that knowledge to pick the answer. How would an intelligent agent do this?

Function of a Cognitive Architecture

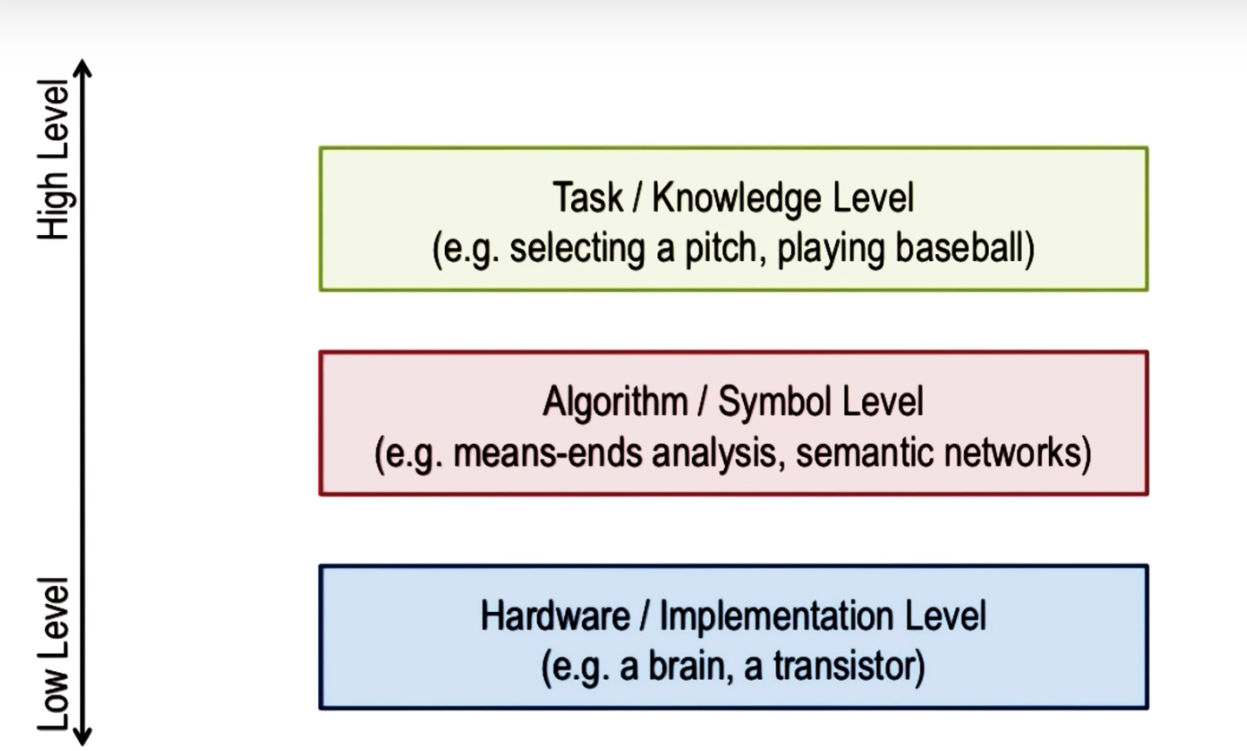
Definition of an agent: a function that maps a perceptual history to an action

f: P* -> A

(History of) Percepts -> Action

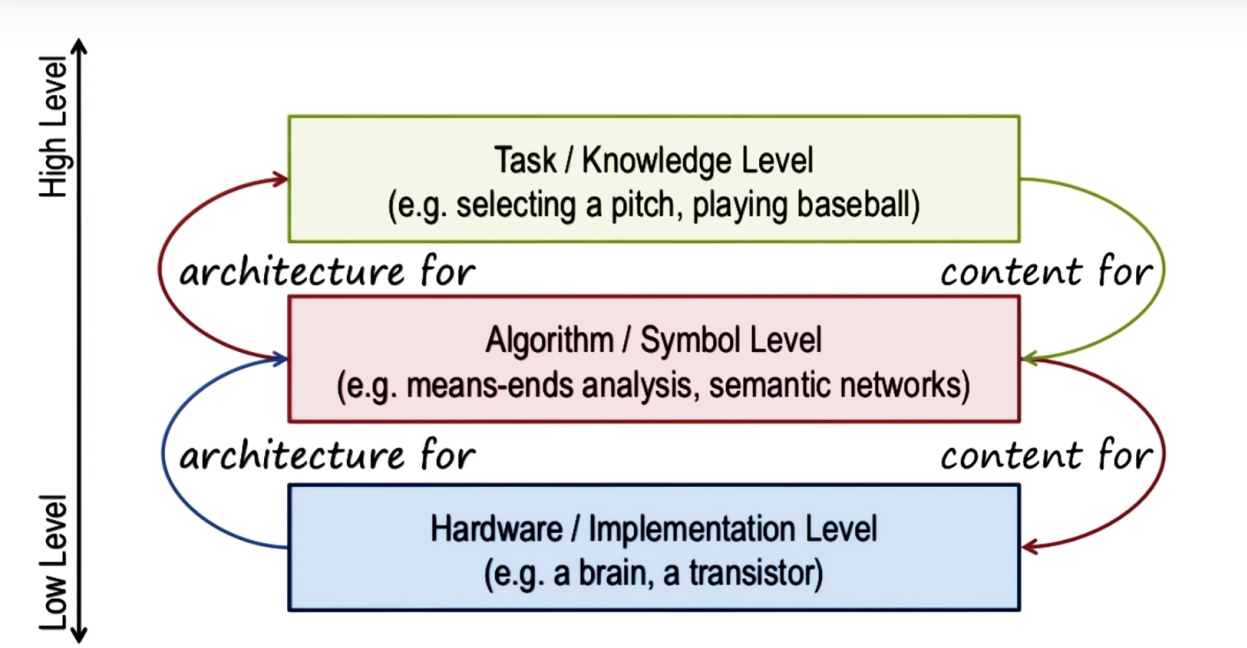
One of the major tasks of cognitive agents is to select actions to take.

Levels of Cognitive Architecture



Various levels are connected with each other. Hardware level is a level for implementing what is happening at the algorithm level. Algorithm provides architecture for task level.

Opposite direction: Task provides content of the knowledge that needs to be manipulated in the algorithm, and the algorithm provides the content for what needs to be processed at the hardware level.



Most of KBAI will be focused around the top two layers of abstraction with some allusions to hardware.

Layers of Analysis for IBM's Watson:

The physical computer (hardware)
Algorithm (searching and decision-making)
Task (answering the inputted clue)

Assumptions of a Cognitive Architecture

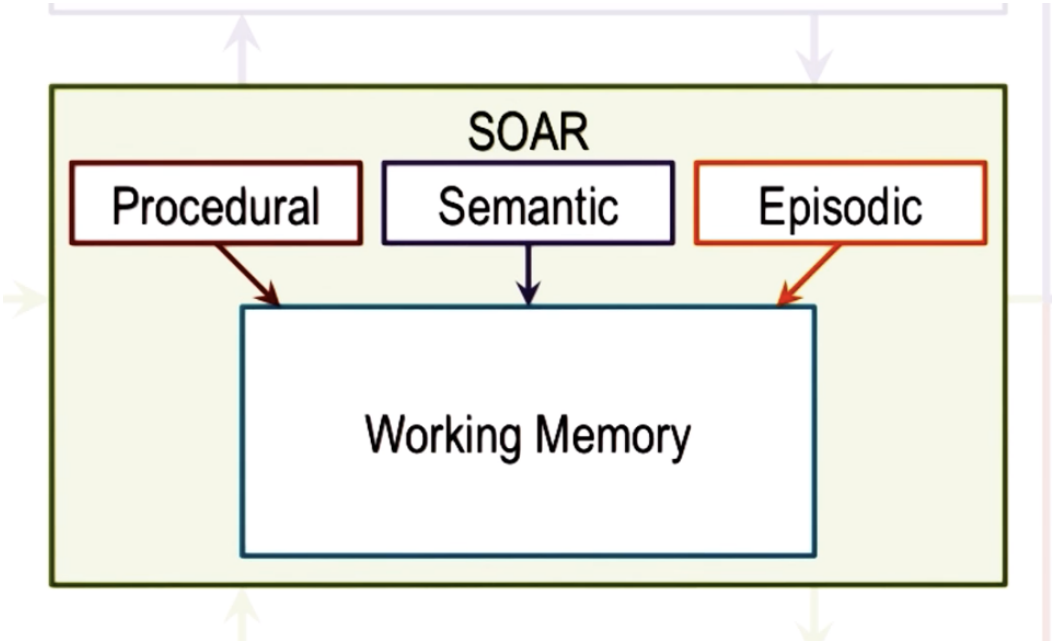
- Goal-oriented or goal-directed
- Live in rich, complex, dynamic environment
- Use knowledge about the world (significant)
- Symbols and abstractions
- Flexible and function of the environment
- Learn from experience

Architecture + Content = Behavior

If the architecture is fixed, we only need to modify the content.
We can map behavior to content because the architecture is fixed.
Keep architecture constant, change content.
So what is a good architecture? That's next.

A Cognitive Architecture for Production Systems

A specific architecture for deliberation: SOAR



SOAR consists of a long term memory and a working memory.
The long term memory contains 3 types of knowledge:

Episodic - events (what did you have for dinner yesterday)
Semantic - generalizations in the form of concepts and models of the world (your model of how a plane flies)
Procedural - how to do certain things (how to pour water from a jug into a tumbler)

Return to the Pitcher

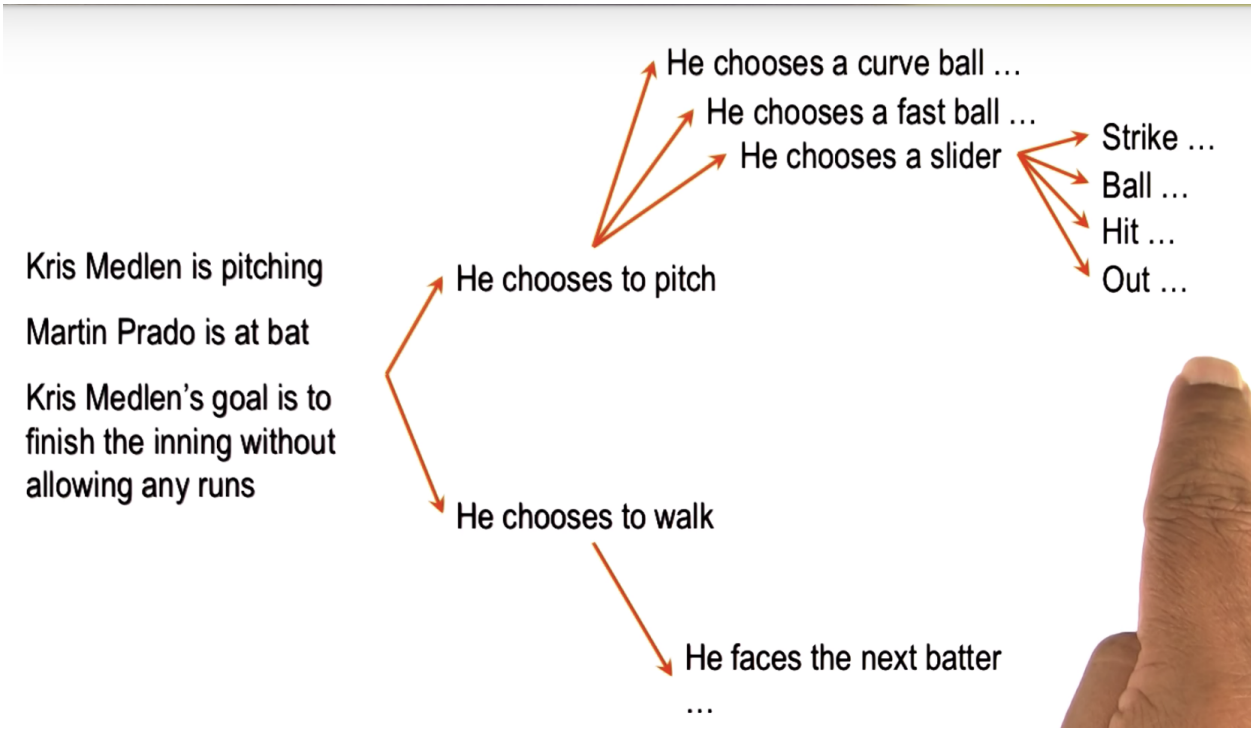
Pitcher has to decide on an action
P* -> A

How will the pitcher as an AI agent come to this decision?
How do the percepts (information) get mapped to the action to walk the batter?
Pitcher has several kinds of knowledge - some internal (goals and objectives), some can be perceived from the environment (such as bases, state of game, batter, score, etc.)

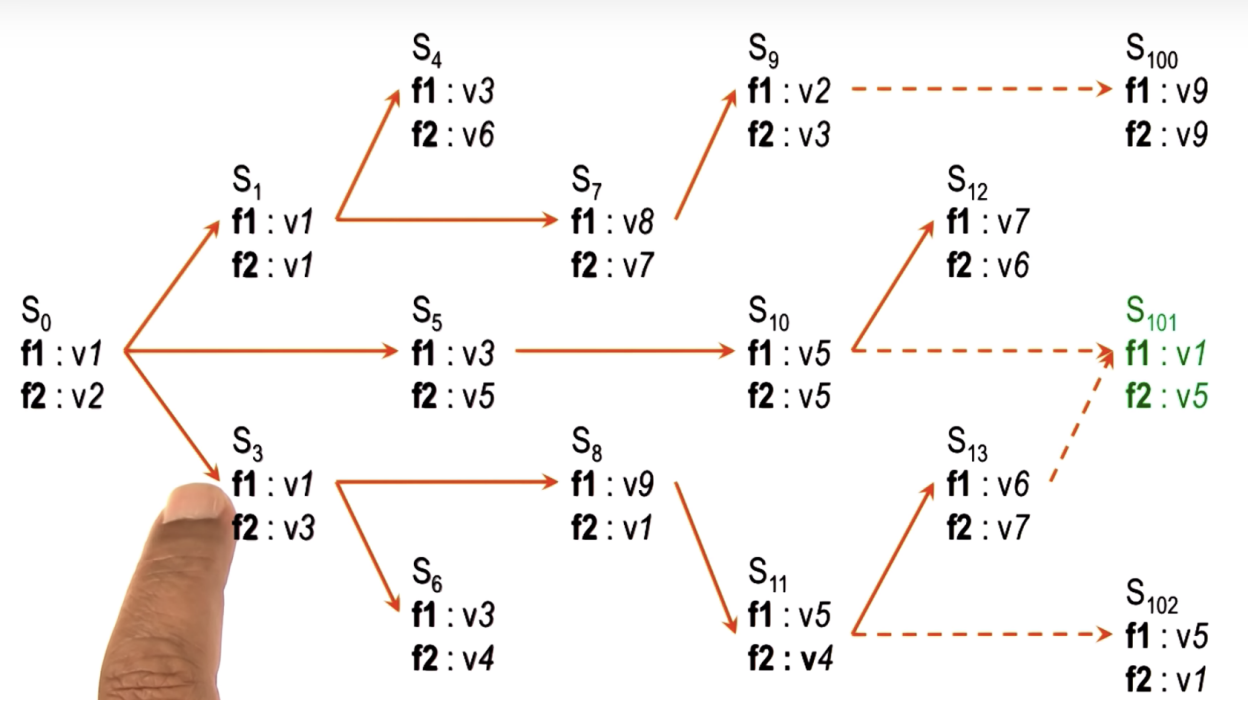
Action Selection

Pitcher may look at various actions available and look at additional possibilities that appear after making each decision. Sets up a state space.

Informal:



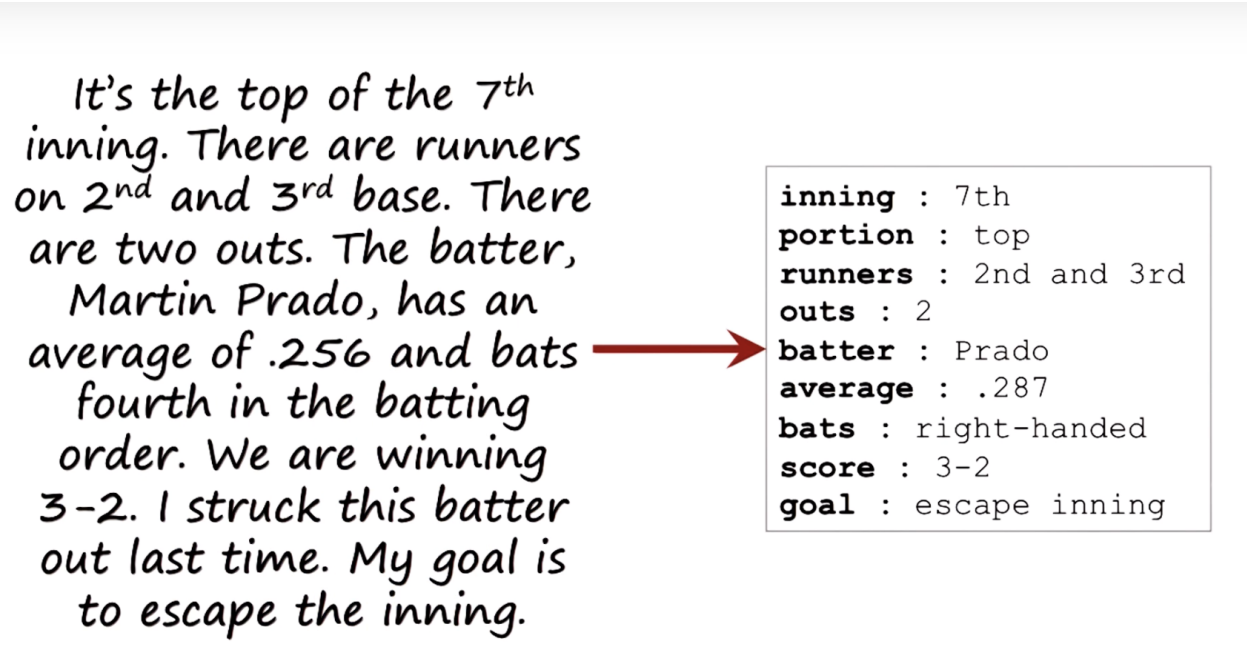
Formal description:



Can think of pitcher's decision as an abstract state space and exploring the state space in an

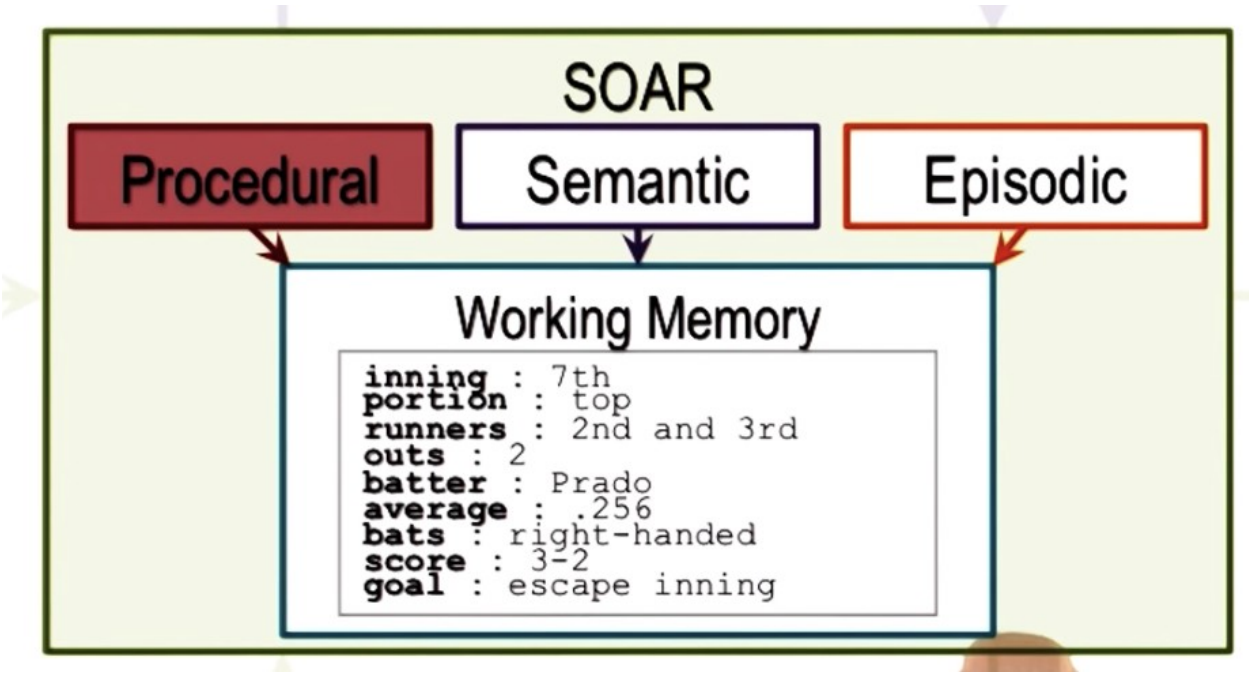
attempt to map a path from the starting state to the goal state.

Putting Content in the Architecture

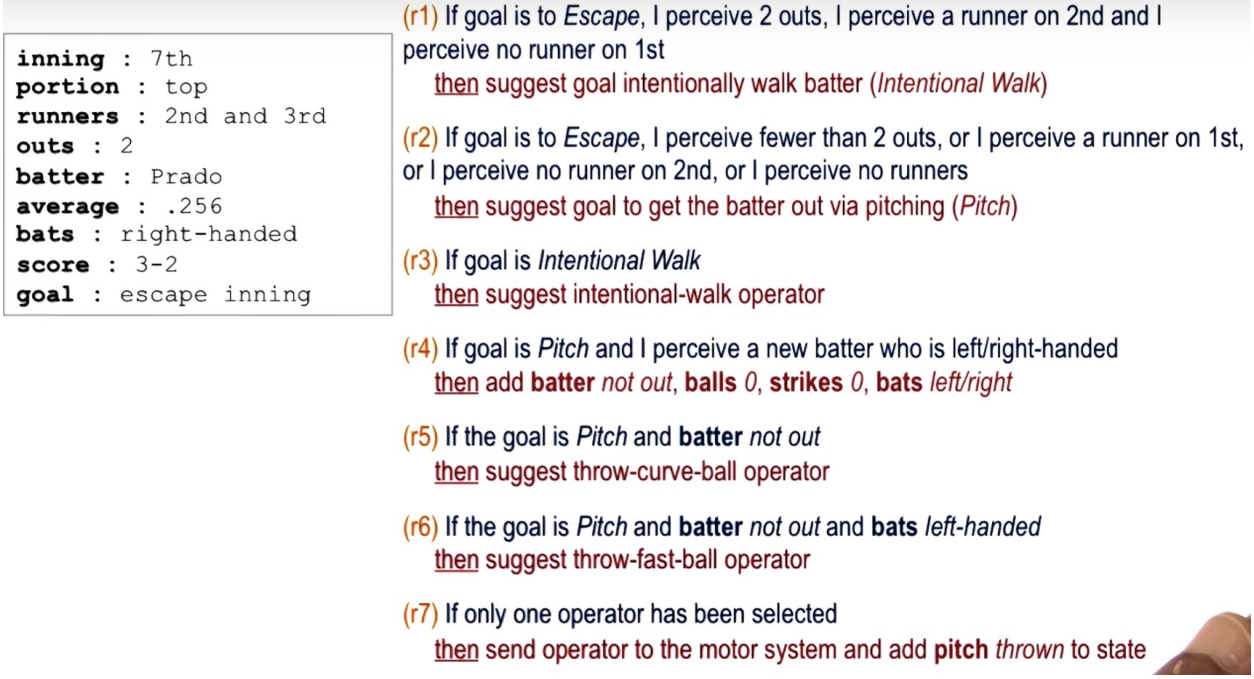


Bringing in Memory

SOAR's working memory now contains all the properties described above that have been mapped like so:



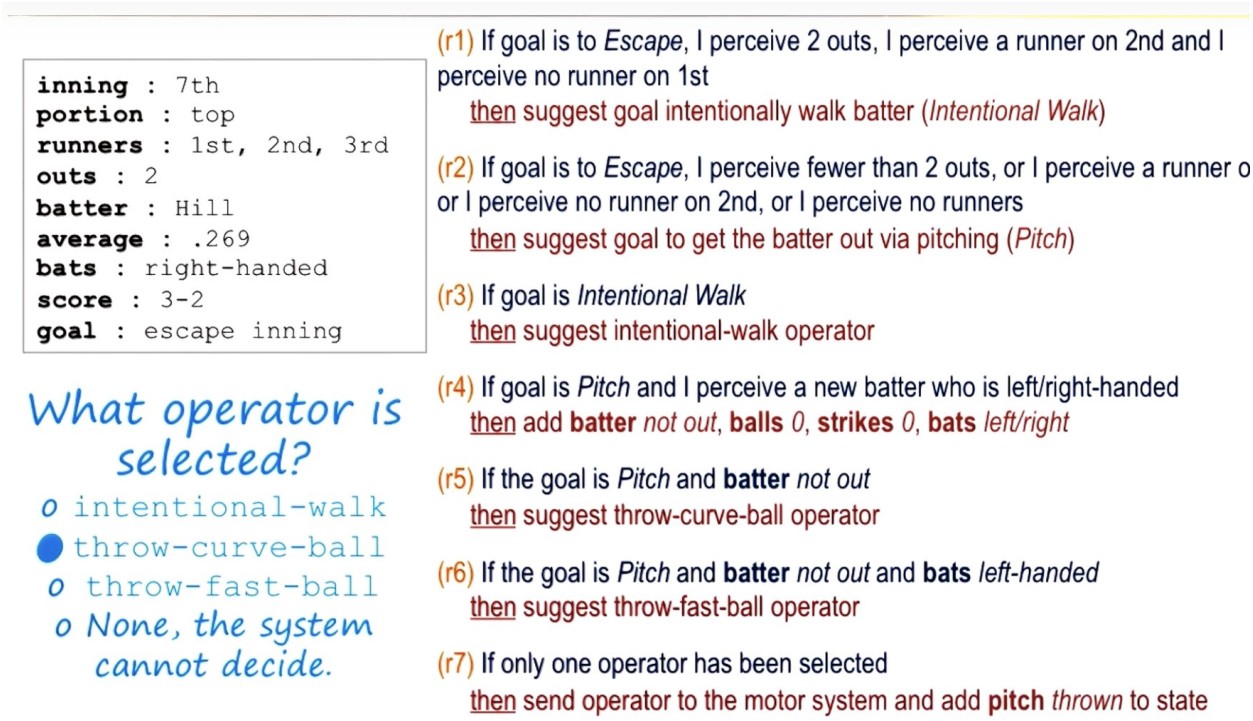
Some are percepts, others are pitcher's internal goals.
Imagine that the procedural part of SOAR's long term memory contains rules. Sometimes called "production rules" (the term "Production Systems" comes from this term). One of the first things the pitcher had to decide was whether to throw a pitch or walk a batter. There are rules to account for this.



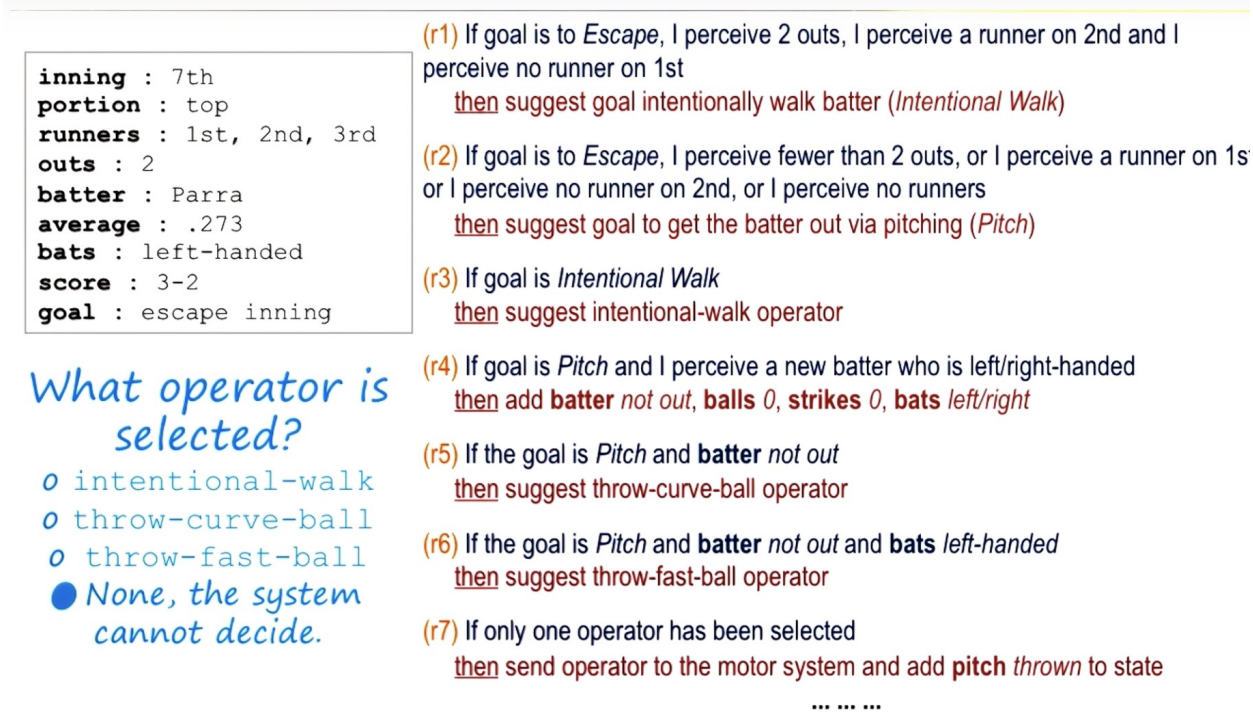
SOAR's long term memory consists of various types of knowledge.
One type is procedural knowledge (about how to do something).

This knowledge is presented in the form of production rules of the form:
If (something) then (something).
There are antecedents and consequences, both can be connected with relationships like “and” and “or”.

Another example where first rule is skipped due to runner on first:



As the contents of the working memory changes as this process activates, new rules can get activated.
This can also lead to confusing situations, such as when a left-handed batter in a similar situation is entered as data. These rules are applied in sequence and are not skipped, so more than one state can be suggested, see below in the case of (r5) and (r6), since (r5) is detected first, it suggests curve ball, but once (r6) is processed, it suggests fast ball. So in this case we don’t fire (r7) because more than one operator was selected! No rule tells us what action to take for multiple results!

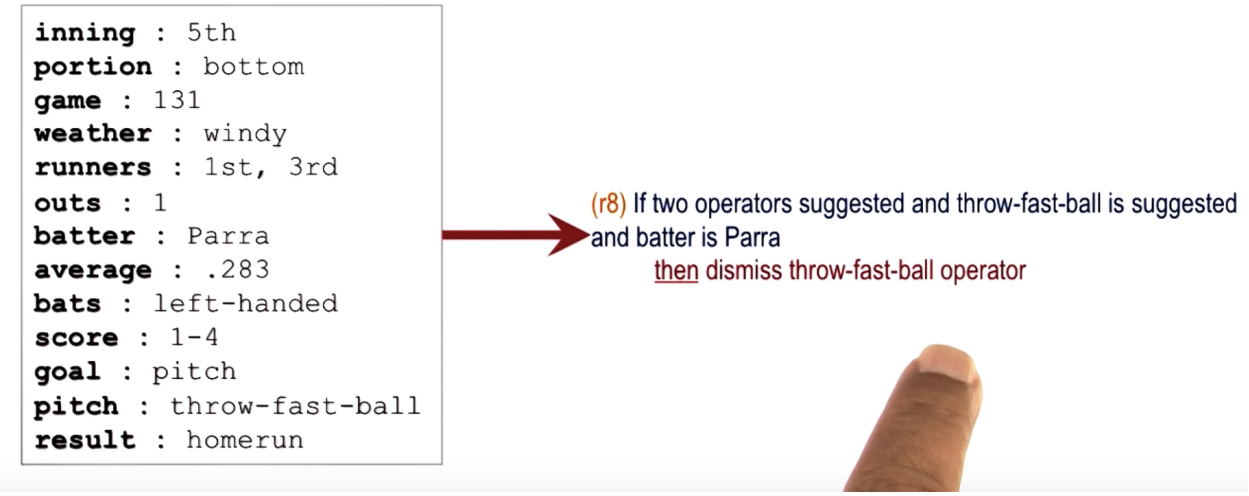


Chunking

In this case SOAR selected 2 actions and the system does not have additional rules with which to decide on an action to take. Should the pitcher throw a fast ball or a curve ball?

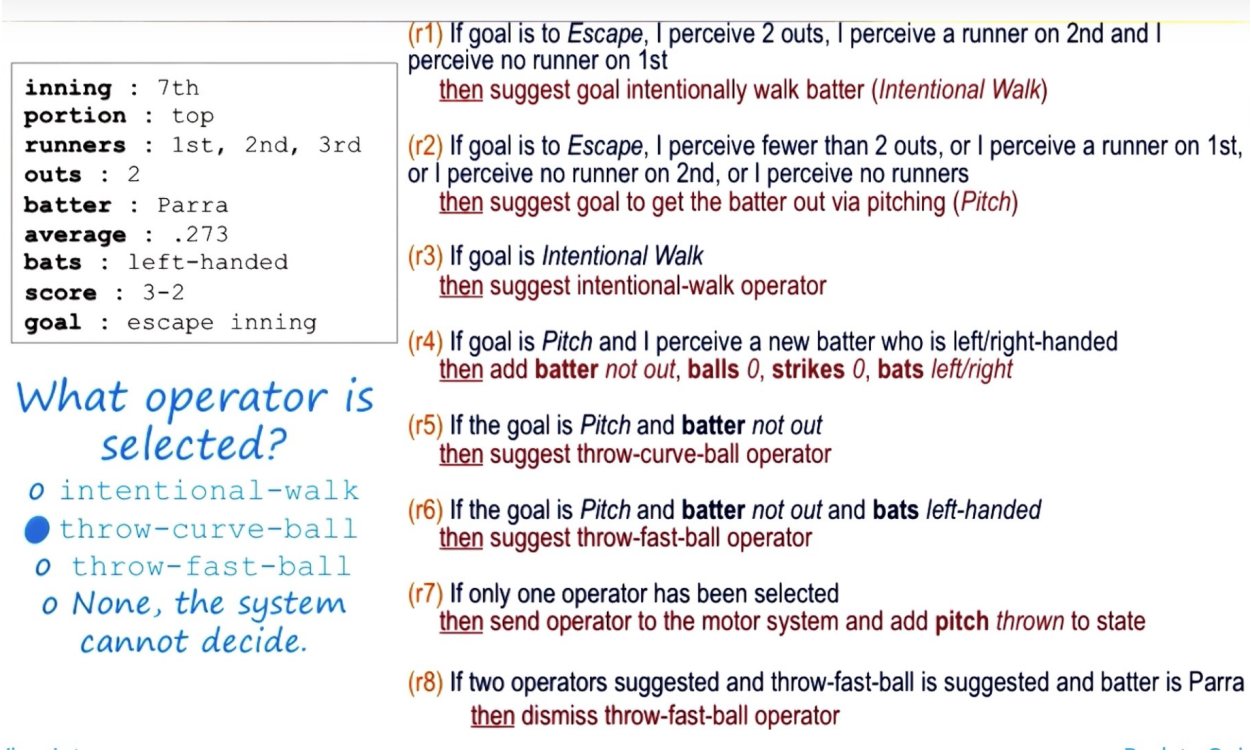
At this point, SOAR will attempt to learn a rule that may break the impasse. If the decision maker has a choice between a fast ball and a curve ball, might there be a way of learning a rule in a particular situation given these choices? SOAR will invoke episodic knowledge in this case.

Imaging SOAR has previous knowledge of a situation that occurred earlier.
SOAR can encapsulate knowledge from episodic event into a production rule that can be used as a part of the procedural knowledge. In this case, we see that a previous event with that batter showed that a fast ball thrown to that same batter led to a homerun, pitcher does not want that, so fast ball should not be thrown.



This is the process of learning known as Chunking.
Chunking is a learning technique that SOAR uses to learn rules that can break impasse.
Chunking is triggered when impasse occurs (two rules are activated and no means of resolving data between them), and the goal is to break the tie.
SOAR searches through episodic memory to find event that has knowledge that could break the impasse (looking at precepts of current situation and finds similar situations in episodic memory). If a match is found, it will take that similar situation's results into consideration when building a rule.

Once adding that additional rule into the problem, SOAR can choose a valid result:



Fundamentals of Learning

How do agents learn?
What do agents learn?
What is the source of their learning?
Why do they learn?

Fundamental stance of KBAI on this: We start with a theory of reasoning that starts with what, when, why to learn.

After that, we can address the question of how.

Cognitive Connection

Production systems were proposed as models of human cognition.
Working memory in production system maps to human short-term memory.
Strong similarities between SOAR and humans when given arithmetic/mathematical problems.

Lesson 7: Frames

A knowledge representation and a first step towards building a theory of common sense reasoning.

- Function of frames

- Properties of frames
- Relationship between frames and previous topics (semantic networks)
- Frames for advanced sense-making

What is common-sense reasoning?
What is the meaning of the sentence: “Ashok ate a frog?”

Is the frog alive or dead?
Where is the frog?
Is Ashok happy or sad?

None of those could be reasoned directly from the sentence above, we as humans reasoned through common sense.

How do we make sense of a sentence?

What is the “meaning” of the meaning of the sentence. We break it apart.
We associate a frame with the verb.

Ate
- Subject: (eater)
- Object: (eaten)
- Location:
- Time:
- Utensils:
- Object-Alive:
- Object-Is: (location of object)

So for this example:

Ate
- Subject: Ashok
- Object: frog
- Location:
- Time:
- Utensils:
- Object-Alive: false
- Object-Is: in-subject
- Subject-Mood: happy

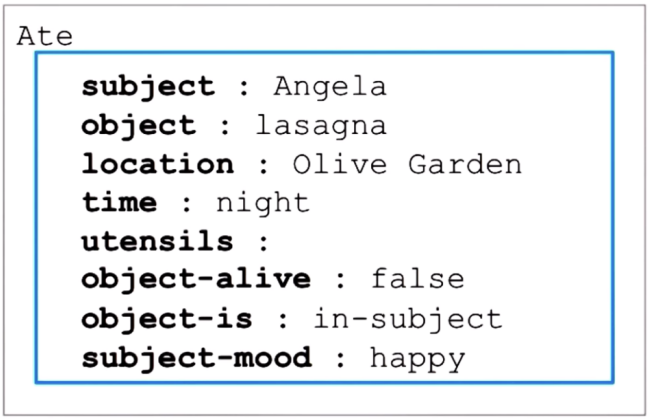
Property names are “slots”. Values are “fillers”.

A frame is a knowledge structure. There are a number of things happening in this knowledge representation.

Some KRs are like atoms (production rules), others are molecules (frames) - they can expand, contract, change.

Making sense of a sentence:

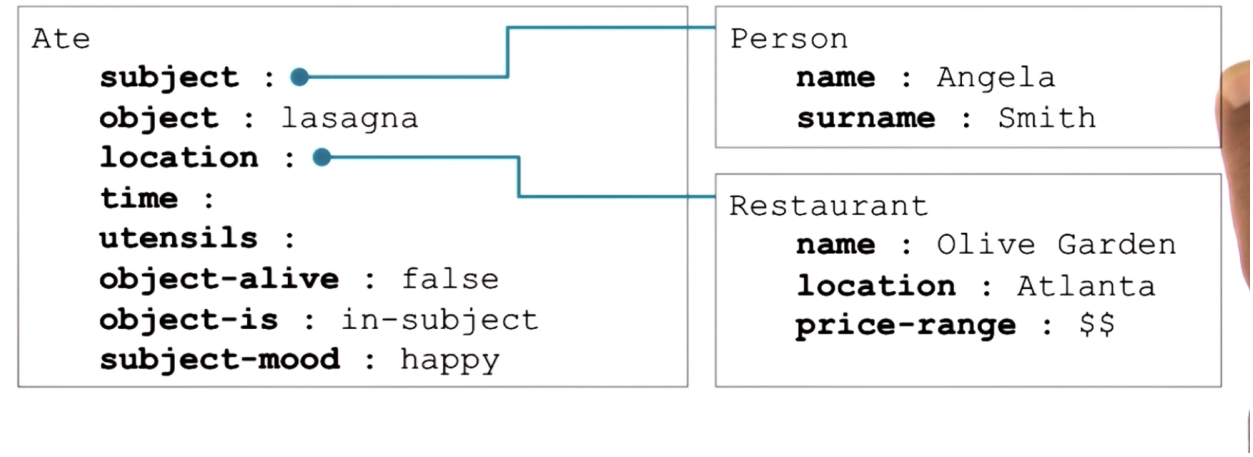
Angela ate lasagna with her dad last night at Olive Garden.



Could have added her dad as a slot, but haven't discussed yet.
Each filler can come from a range of values.
Common sense reasoning can be programmed using default values for fillers (e.g. object-alive false by default)

Complex Frame Systems

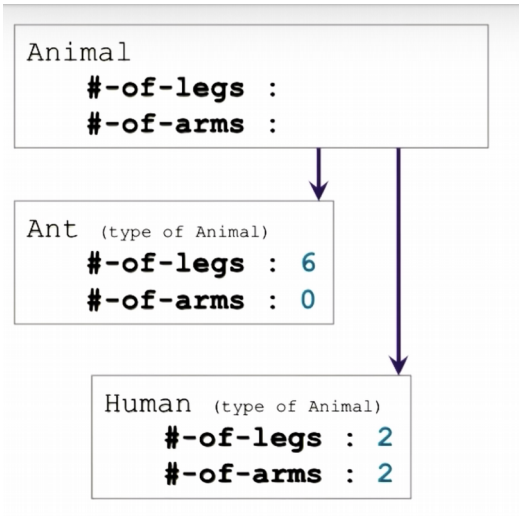
Aside from verbs, frames can also be used for nouns.



Frames can be connected with one another so that fillers can point to different frames. When frames are hooked up together, we can start building out larger concepts of understanding using this model.

Properties of Frames

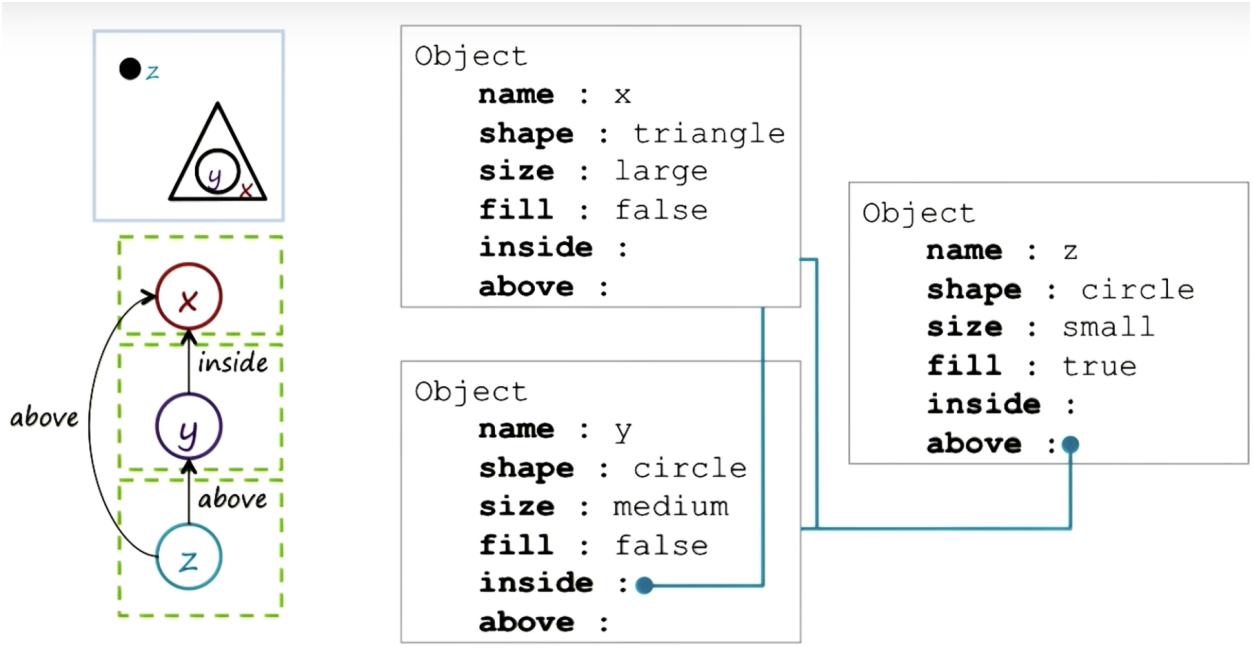
- Frames represent stereotypes (slots are defined by stereotypical representation)
- Frames provide default values that can be overridden. It's both powerful and a problem, it's difficult to manage exceptions as problem space grows.
- Frames exhibit inheritance (in a hierarchy), just like classes in OOP



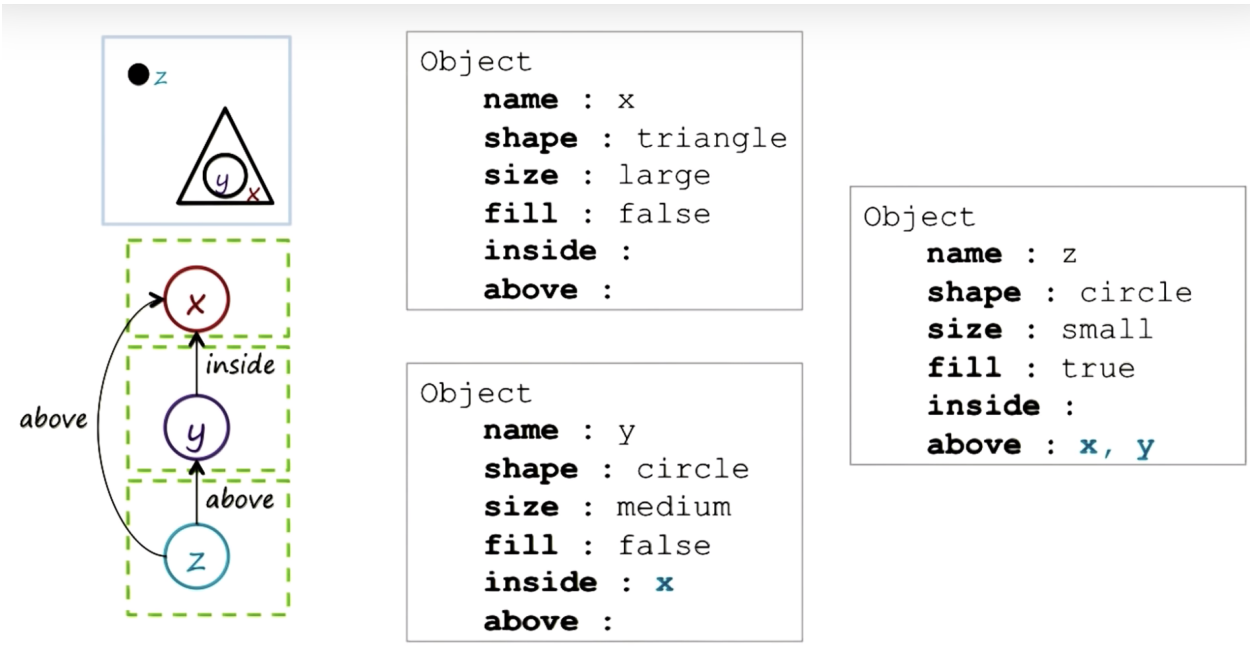
Default values concept in frames is similar to constructor in a class in OOP. Frames and OOP came about during the same era and have many influences.

Frames and Semantic Networks

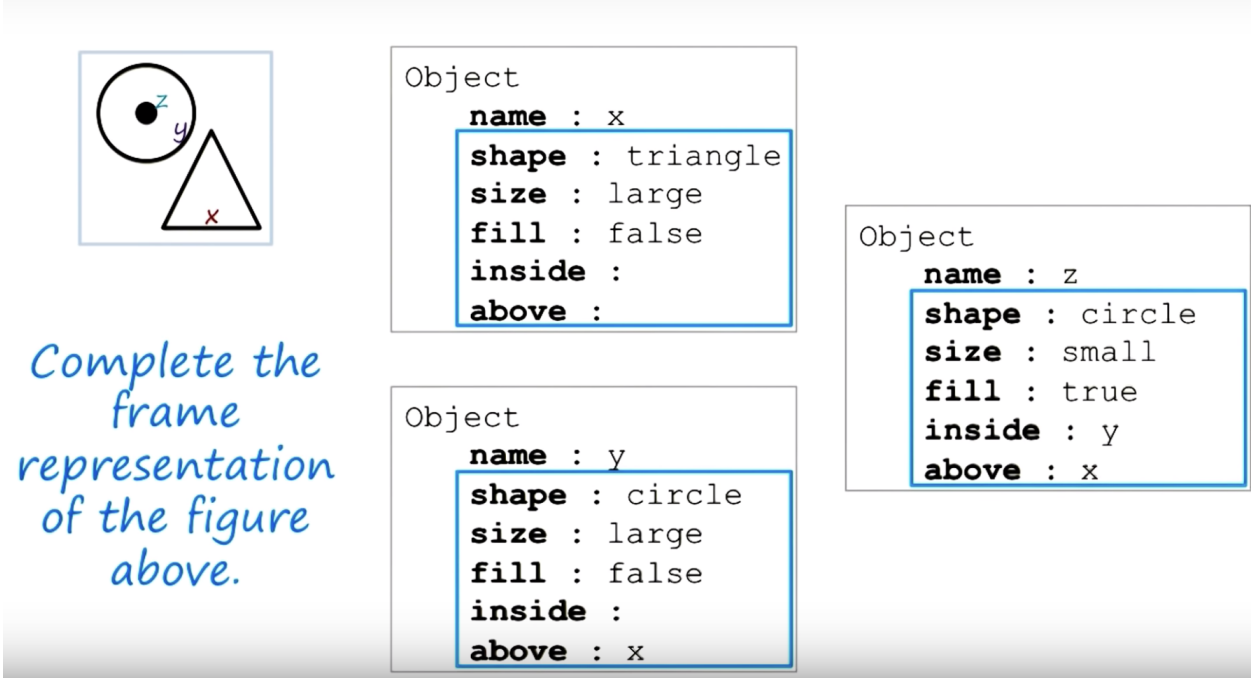
Can also address Raven's problems using Frames.
Frames and semantic networks are closely related.
A semantic network can be rewritten in the language of frames.



Relationships between frames are captured by blue lines, or through specifying variables or other frame names (see z, y and x below).

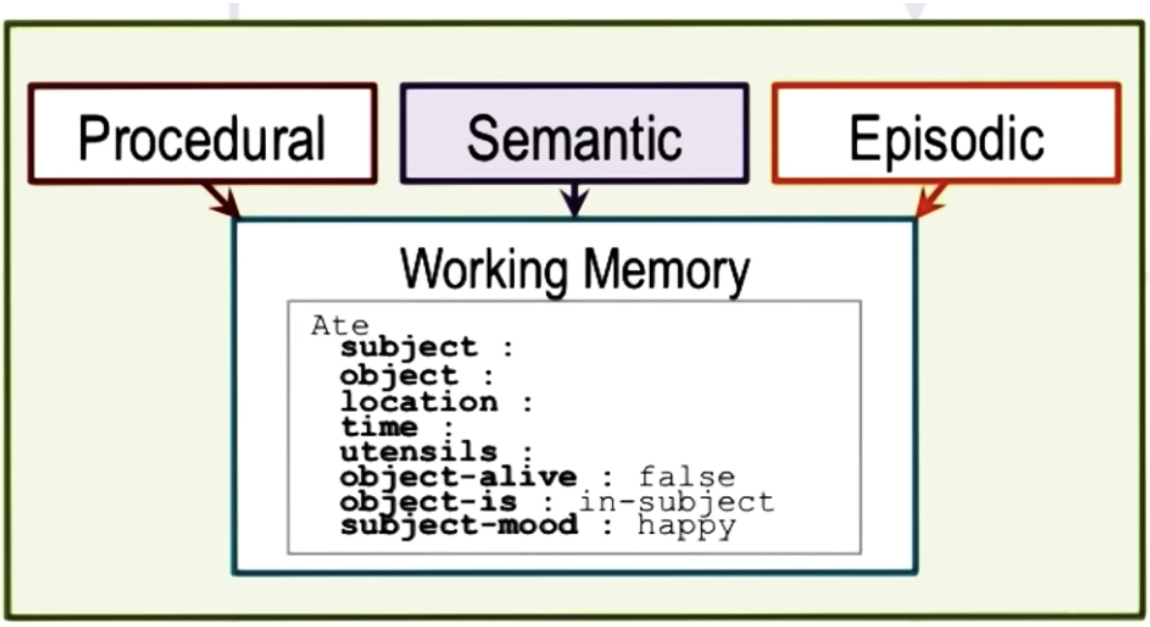


Another example of frame representation of a Raven's image:



Frames and Production Systems

In the working memory container of SOAR, a frame can be used used!



Another example, describing a frame representation of an earthquake:

Today, an extremely serious earthquake of magnitude 8.5 hit Lower Slabovia, killing 25 people and causing $500 million in damage. The President of Lower Slabovia said that the hard-hit area near the Sadie Hawkins fault has been a danger zone for years.

Write a frame representation of this story on the right.

Earthquake

day : Today
location : lower Slabovia
damage : $500 million
fatalities : 25
faultline : Sadie Hawkins
magnitude : 8.5
time :
type :
duration :

Limitations of language understanding can create issues when analyzing two stories about different outcomes. How does it know Sadie Hawkins is a science advisor vs. a faoult, or killing proposals vs. killing people?

Today, an extremely serious earthquake of magnitude 8.5 hit Lower Slabovia, killing 25 people and causing $500 million in damage. The President of Lower Slabovia said that the hard-hit area near the Sadie Hawkins fault has been a danger zone for years.

Today, the **President of Lower Slabovia killed 25** proposals totaling **$500 million** for research in **earthquake** prediction. Our Lower Slabovian correspondent calculates that **8.5** research proposals are rejected for every one approved. There are rumors that the President's science advisor, **Sadie Hawkins**, is at **fault**.

The Cognitive Connection

3 specific ways they map to human cognition:

Frames are a structured knowledge representation (molecule, not atom).
Frames enable you to generate a theory of cognitive processing both bottom-up and top-down.
Frames capture the notion of stereotypes, though stereotypes can lead to incorrect inferences as well. Stereotypes are cognitively efficient. Default values. Property of frames too.

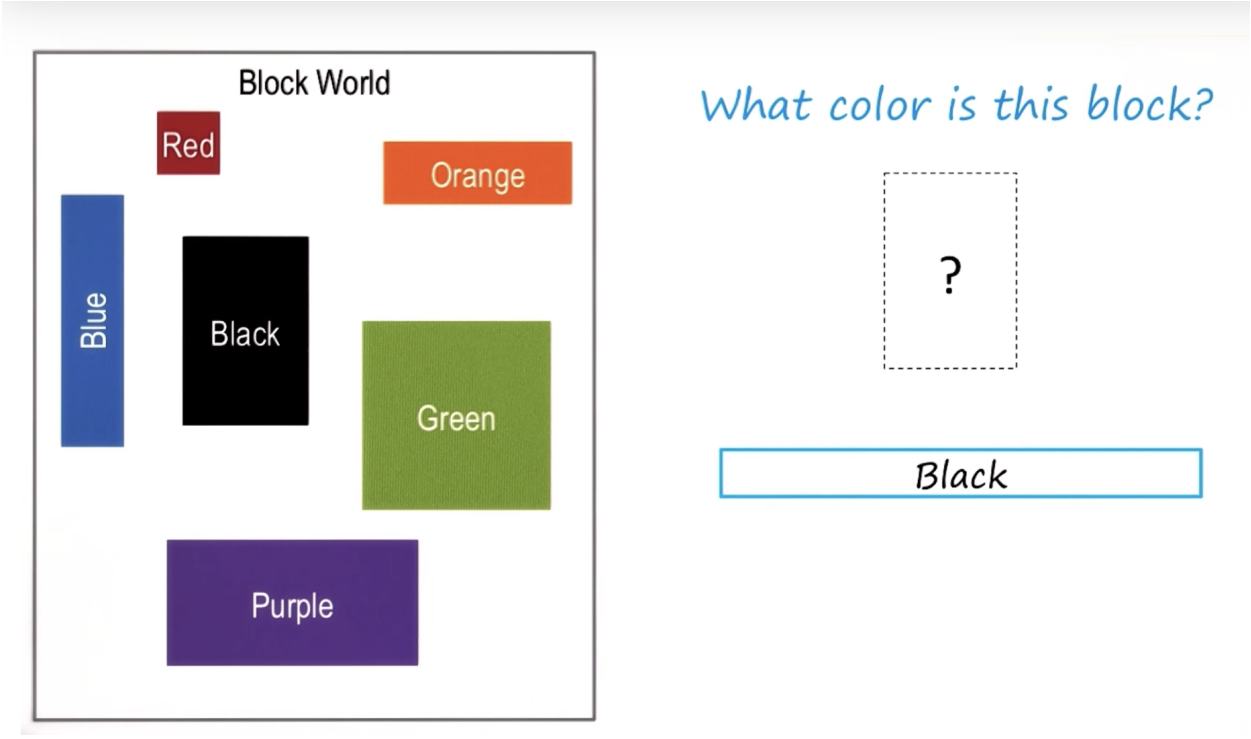
Lesson 8: Learning by Recording Cases

First topic in analogical reasoning

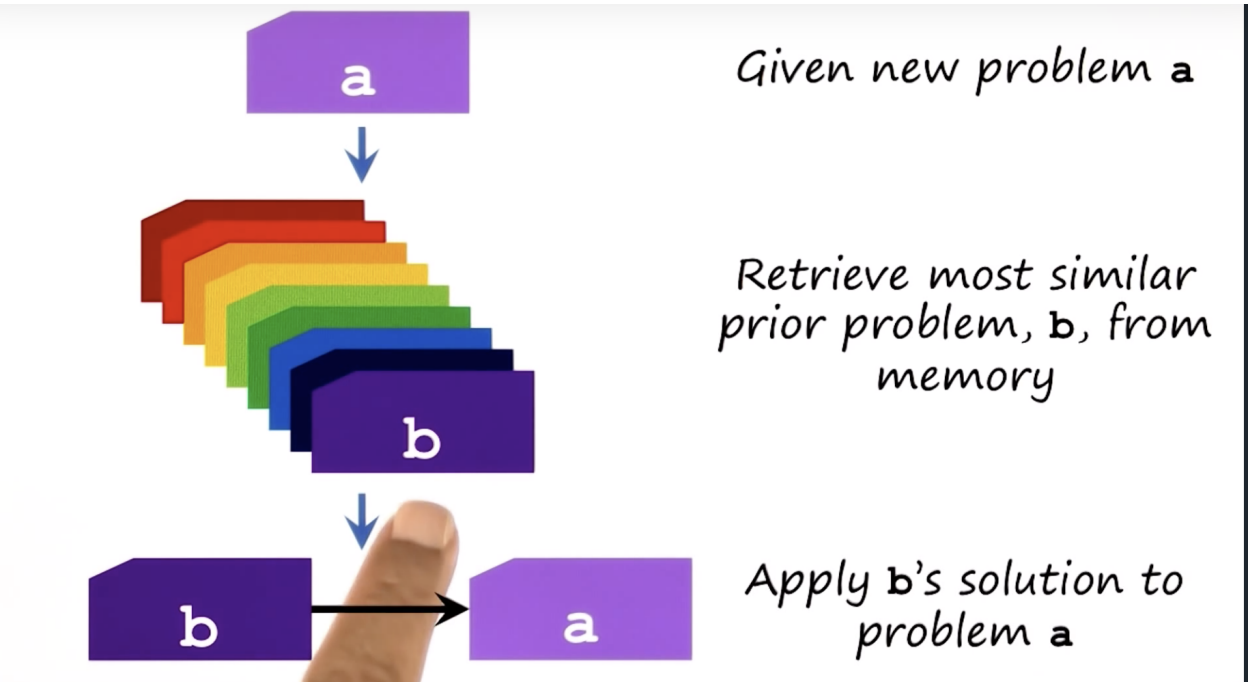
- Learning by recording cases
- Nearest neighbor
- k-Nearest neighbor
- Complex cases in the real world

Block World

Consider a world of blocks colored of various shapes and sizes.
Based on experience in this world, what is the color of the block shown?
Black - based on shape, size, and orientation.



This is an example of learning by recording cases. 6 cases were recorded in agent's memory (the existing blocks), then the agent will answer a new problem based on cases in memory.



Example: programming patterns - new problem, but applying similar solution.

Case is an encapsulation of a previous experience. Works in a very large number of situations from tying shoelaces on new shoes to medical diagnoses.

Case Retrieval by Nearest Neighbor

How can we make this more explicit?

We can represent the problem of block world by a 2D matrix, height x width.

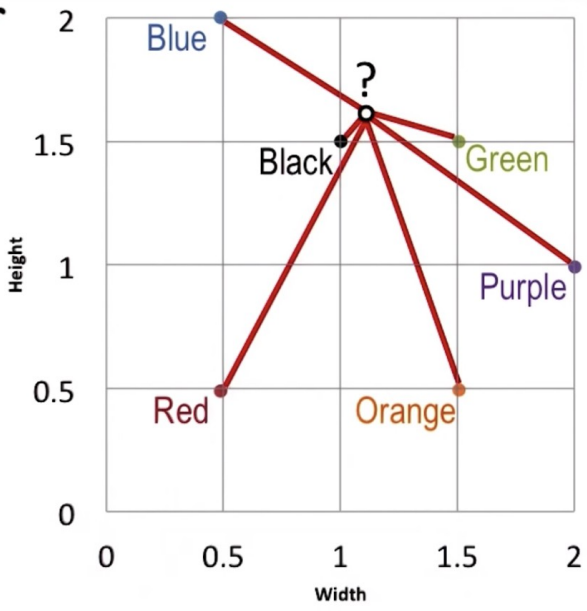
One measure to determine how “close” a solution would be to a new problem is Euclidean distance. We compute all Euclidean distances from the location of the new problem on the grid to all other problems and quickly discover the shortest distance to the Black block.

Finding the Nearest Neighbor

Given existing case at (x_c, y_c)
and new problem at (x_n, y_n)

$$d = \sqrt{(y_c - y_n)^2 + (x_c - x_n)^2}$$

Block	x_c	y_c	x_n	y_n	d
Blue	0.5	2.0	1.1	1.6	0.72
Red	0.5	0.5	1.1	1.6	1.25
Black	**1.0**	**1.5**	**1.1**	**1.6**	**0.14**
Green	1.5	1.5	1.1	1.6	0.41
Orange	1.5	0.5	1.1	1.6	1.17
Purple	2.0	1.0	1.1	1.6	1.08



Recording Cases in Real Life

Given a map of a small portion of Long Island, NY.
An automatic car can navigate different routes.
Car only navigates via learning by recording cases.
New problem: how to go from Q to arrow?
We need to look at both the possible solution's origin as well as where the route ends up!

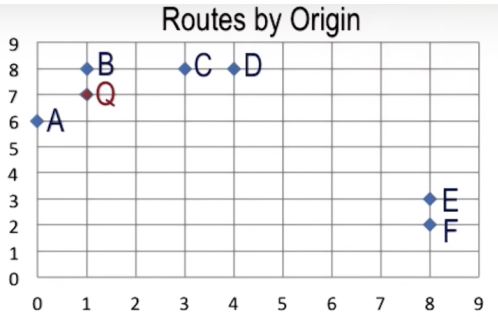
What route is most similar to this new problem?

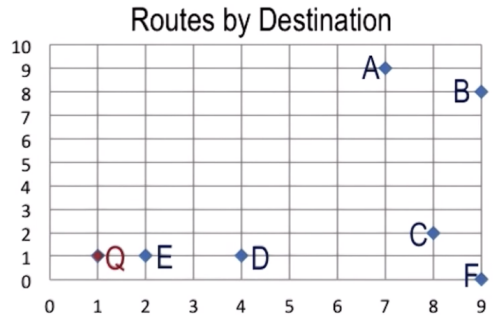
How can we program an AI agent to come up with this answer?

Nearest Neighbor for Complex Problems

Routes by Origin and Routes by Destination with Euclidean Distance

	Origin			Destination		
Route	x_o	y_o	d_o	x_d	y_d	d_d
A	0	6	1.41	7	9	10.00
B	**1**	**8**	**1.00**	9	8	10.63
C	3	8	2.24	8	2	7.07
D	4	8	3.16	4	1	3.00
E	8	3	8.06	**2**	**1**	**1.00**
F	8	2	8.60	9	0	8.06
Q	1	7	-	1	1	-





Nearest Neighbor in k-Dimensional Space

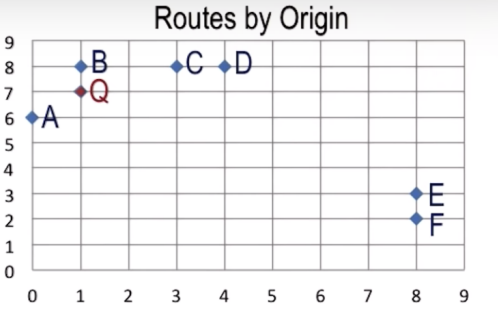
Finding the Nearest Neighbor

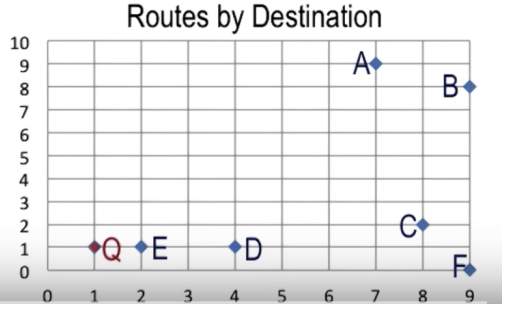
Given existing case at (x_c, y_c)
and new problem at (x_n, y_n)

$$d = \sqrt{(y_c - y_n)^2 + (x_c - x_n)^2}$$

Given existing case at $(c_1, c_2 \ldots c_k)$
and new problem at $(p_1, p_2 \ldots p_k)$

$$d = \sqrt{\sum_{i=1}^{k} (c_i - p_i)^2}$$

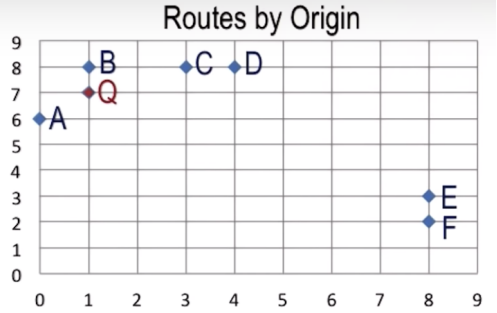




C1 = X(origin)
C2 = Y(origin)
C3 = X(destination)
C4 = Y(destination)

Let's compute one of these.
sqrt((0 - 1)^2 + (6 - 7)^2 + (7-1)^2 + (9-1)^2)
1 + 1+ 36 + 64
sqrt(102) = 10.10 (rounded up) It checks out!

Route	c_1	c_2	c_3	c_4	d_k
A	0	6	7	9	10.10
B	1	8	9	8	10.68
C	3	8	8	2	7.42
D	**4**	**8**	**4**	**1**	**4.36**
E	8	3	2	1	8.12
F	8	2	9	0	11.80
Q	1	7	1	1	-



Mmmm. Alphabet soup is tasty.

However, limitations are present.
Number of dimensions in which new cases need to be computed could be very large. Deciding which case is closest to new problem might be made harder.
Also, even if new problem is close to existing case, doesn't mean the solution of the existing case will solve the new problem.

We'll need methods to fit new cases to the requirements of the new problem (case-based reasoning, will be discussed next lesson).

Cognitive Connection

Learning by storing cases in memory has a very strong connection to cognition.
Cognitive agents are situated in the world. Interactions have certain patterns of regularity.
Problems occurring on a daily basis repeat themselves and memory supplies the answers.

Reasoning, Memory, and Learning.

Intelligence mainly focuses on reasoning/decision-making. But learning by recording cases shifts balance between components and emphasizes memory and learning to decrease necessity to reason as much as we think we need to.

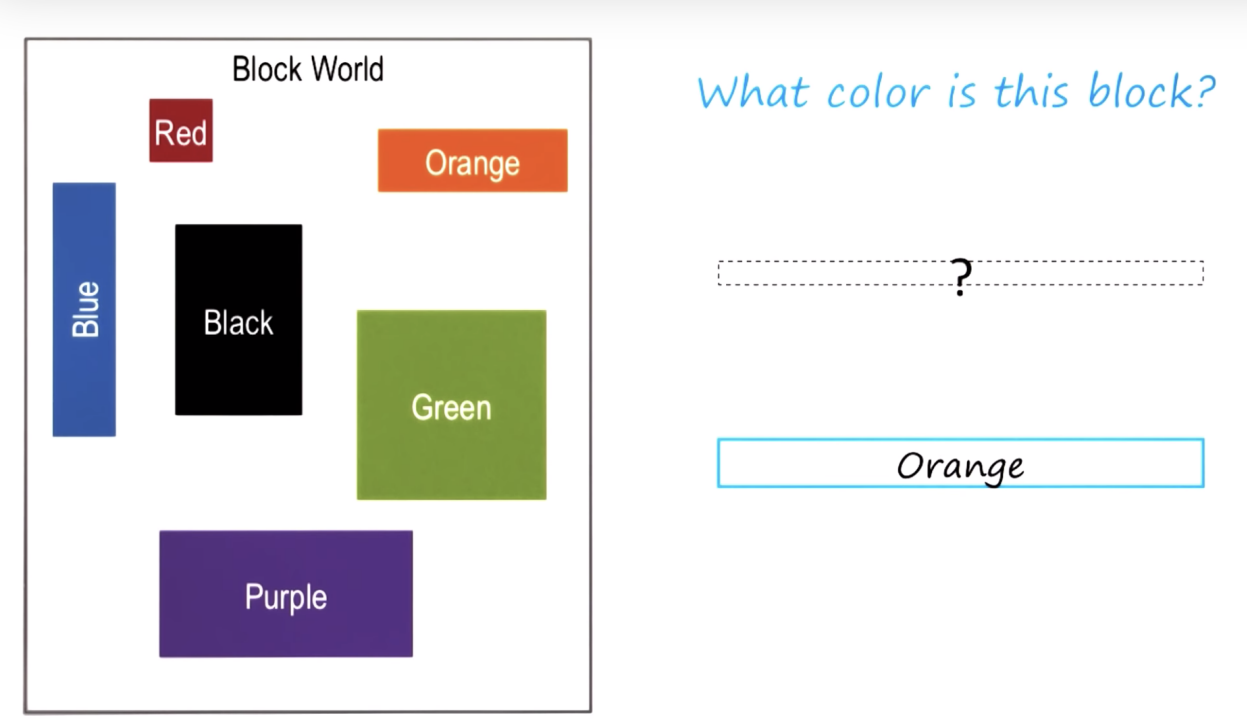
Lesson 9: Case-Based Reasoning

Cognitive agents addresses new problems by tweaking solutions to similar previously encountered problems. Builds on learning by recording cases (when new problem was identical to previous problem).

- Recording cases revisited
- Need for case-based reasoning
- Phases of case-based reasoning
- Advanced case-based reasoning

Return to Block World

To illustrate difference between case-based reasoning and learning by recording cases, see the problem below. Note that there is no exact match.



Recording Cases to Case-Based Reasoning

Retrieval: Retrieving a case from memory similar to current problem (nearest neighbor)
Adaptation: Adapting the solution to the case to fit the current problem
Evaluation: Evaluate how well the adapted solution addresses the current problem
Storage: Storing the new problem and solution as a case that can be used for future solutioning

Assumptions of Case-Based Reasoning

- Patterns exist in the world (same kinds of problems occur again and again)
- Similar problems often have similar solutions
    - Some exceptions: tying shoes with velcro strap, multi vs single-touch screens

Case Adaptation

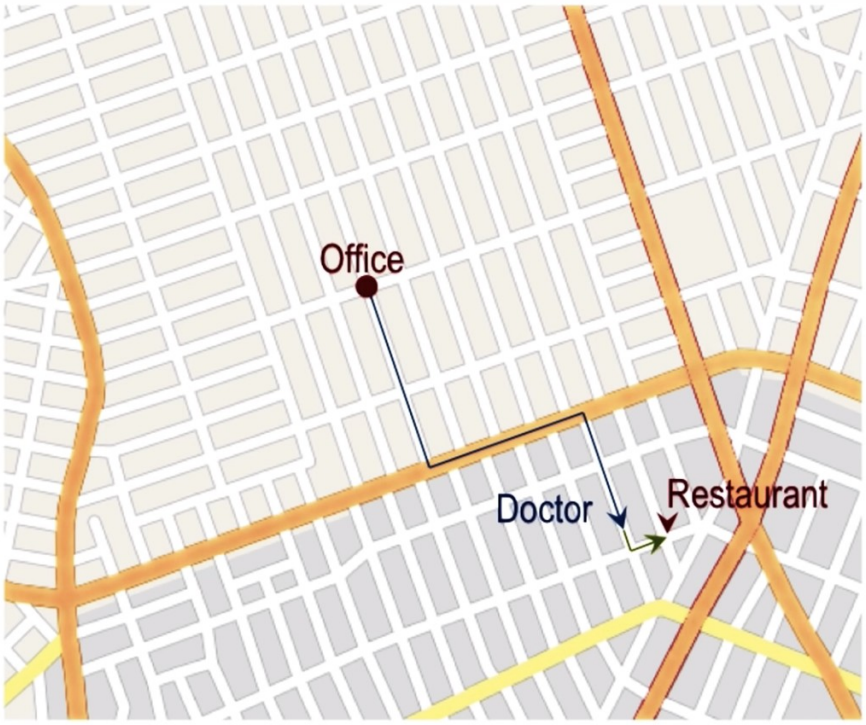
How can we address computationally complex problems with limited cognitive resources in real time? Part of the answer may be that memory supplies the answer (almost-correct answer) so adaptation is very small.

Consider cooking meal for different company than usual, you may tweak for specific people. Or programming example, input data from a file that may be formatted in different ways. All design is redesign.

How can we adapt an older case to meet requirements of the new problem?

Model-based
Recursive case-based
Rules method

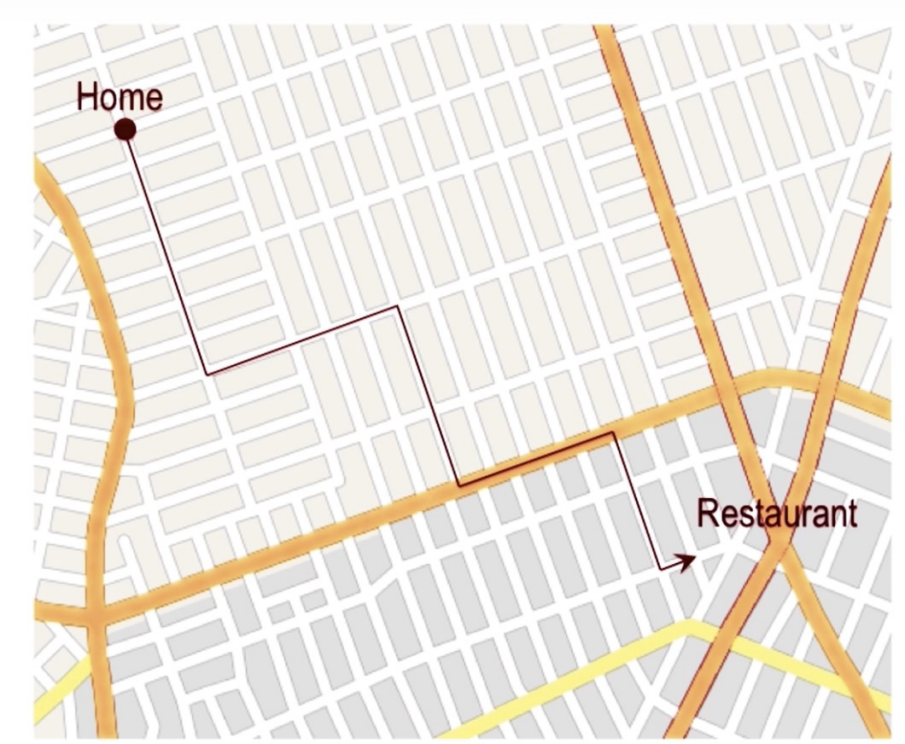
Case adaptation by Model of the World



Assume we begin from office and are trying to get to a restaurant.
We retrieve a case from memory that takes us to a Dr's office close to the restaurant.
We can do a search using this model of the world, finding the restaurant from the Dr's office.

In a programming example, we might have an API that is designed for interacting with a language. We may have the ability to read a file in python but not Java. We can map the functions based on our knowledge of Java to find the solution.

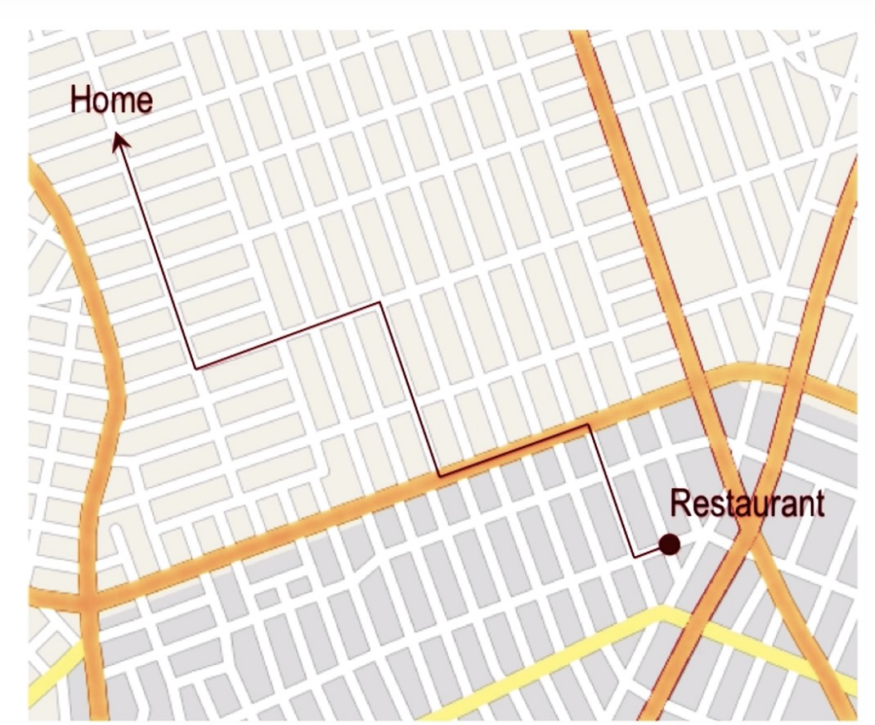
Case adaptation by Recursive Reasoning



Assume we are going from home to the same restaurant.
We know how to go from home to office. And from last time, we know how to get from the office to the restaurant. So we combine both partial solutions.

In programming file example, the eventual goal is to persist data between instances of the program - therefore we can break down file input into one problem, and file output into another problem, both from programs that have been written previously.

Case Adaptation By Rules



Uses heuristics expressed in the form of rules.
How would you find “downtown” in a new town? You could look for tallest buildings (in North America, the tallest buildings are located in that area).
Assume we have to get home from the restaurant.
Now when we have to go back from the restaurant to home, we have a heuristic that allows us to flip back all the turns -- in THIS case.

In programming, file input is often resource intensive, so let’s say for the new program we want to be efficient, so we can make a rule that says it would be better to read full arrays of data at a time instead of one byte at a time.

Heuristics / rules will change between problems.

Case Evaluation

How to assess the suitability of the candidate solution to problem at hand?
We can do a simulation of the found solution, or execute the solution and test it to see if it works.

But that depends on cost of execution!

Every time we run a program and see if it works, we switch back to the adaptation phase.

Case Storage

Important way of learning by accumulating and assimilating new cases.

Case Storage by Indexing

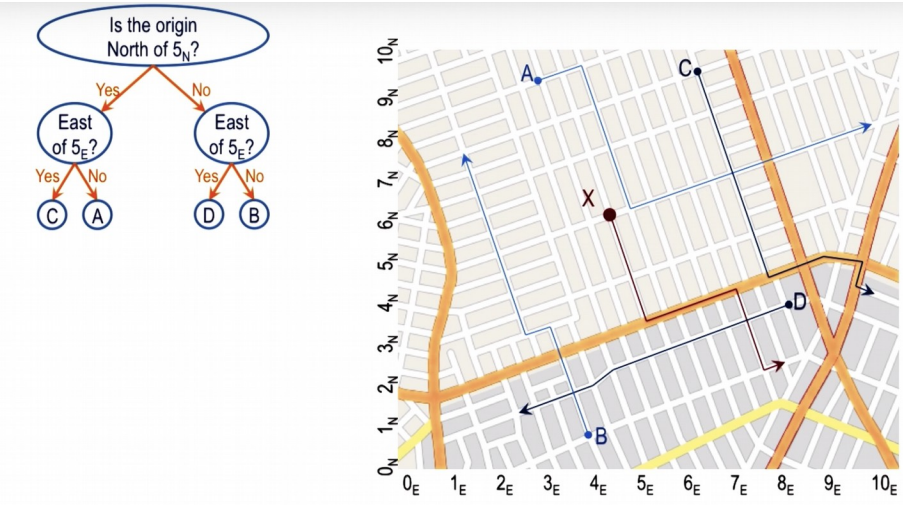
Imagine that we already have cases A, B, C, and D.
We can simply index each solution based on its initial location.

Route	$Origin_X$	$Origin_Y$
A	3_E	9_N
B	4_E	1_N
C	7_E	9_N
D	8_E	4_N
X	**4_E**	**6_N**

But we should take into account destination as well.
We don't have to limit ourselves to just x,y location of origin.
We can add various parameters to the solution (scenic, not scenic), or including destination coordinates of each solution too. We would then determine similarity by computing what solution has the MOST parameters similar to the new problem.
Note that as the number of entries and dimensions increases, retrieval becomes inefficient.

Case Storage By Discrimination Tree

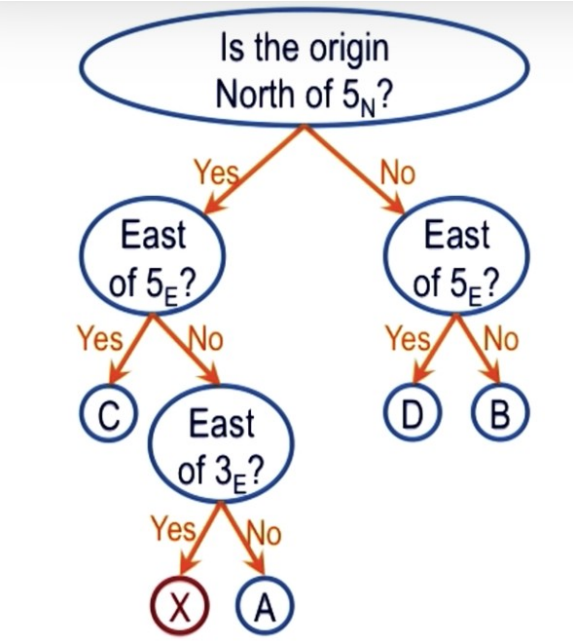
Discrimination tree is a knowledge structure in which the cases themselves are the leaf nodes of the tree. At root and intermediate nodes are questions.



Imagine a new case, X.

If we follow this tree, it would show that X is north of 5N but not east of 5e, forcing it to A's position. We need to somehow distinguish between A and X.

Therefore, we add a new question, say (East of 3E) and refactor both A and X.



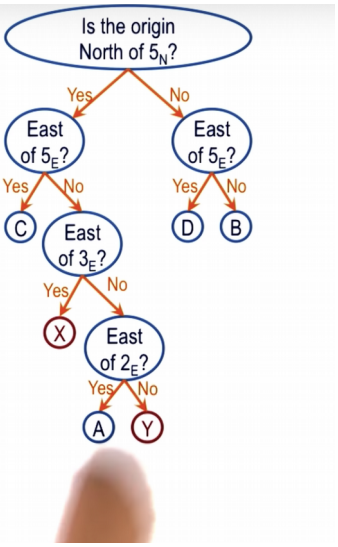
This is not necessarily a binary tree. In block world example, the root node could have a question asking the color of the block, and all child nodes could be expanded that way.

Case Retrieval Revisited

2 Different Ways of Organizing Case Memory:

Tabular and Discrimination Tree.
How can we retrieve a case from memory?
New problem should have the same features in its description as the case in question.
We'll use the problem to navigate the tree and find out which case is most similar.

Route	$Destination_X$	$Destination_Y$
A	10_E	8_N
B	1_E	8_N
C	10_E	4_N
D	2_E	1_N
X	8_E	2_N
Y	8_E	2_N
Z	1_E	9_N



Advanced Case-Based Reasoning

If evaluation found and solution failed, try adapting again
If evaluation found and solution failed, try retrieving a different solution
The retrieved solution could not be adapted, retrieve a different solution,
Retrieved case perfectly matching new problem; no adaptation needed

What about evaluation showing failed cases? Should we store those?
Sometimes - they can help anticipate the kinds of issues that will occur when solving new problem.
Failures are great opportunities for learning.

Cognitive Connection

Analogical reasoning depends on a spectrum of similarity; at one end are problems identical to others, at the other end are problems not similar at all. In the middle are problems that are similar but not identical - those we can retrieve solutions, tweak them, and apply those to new problems.