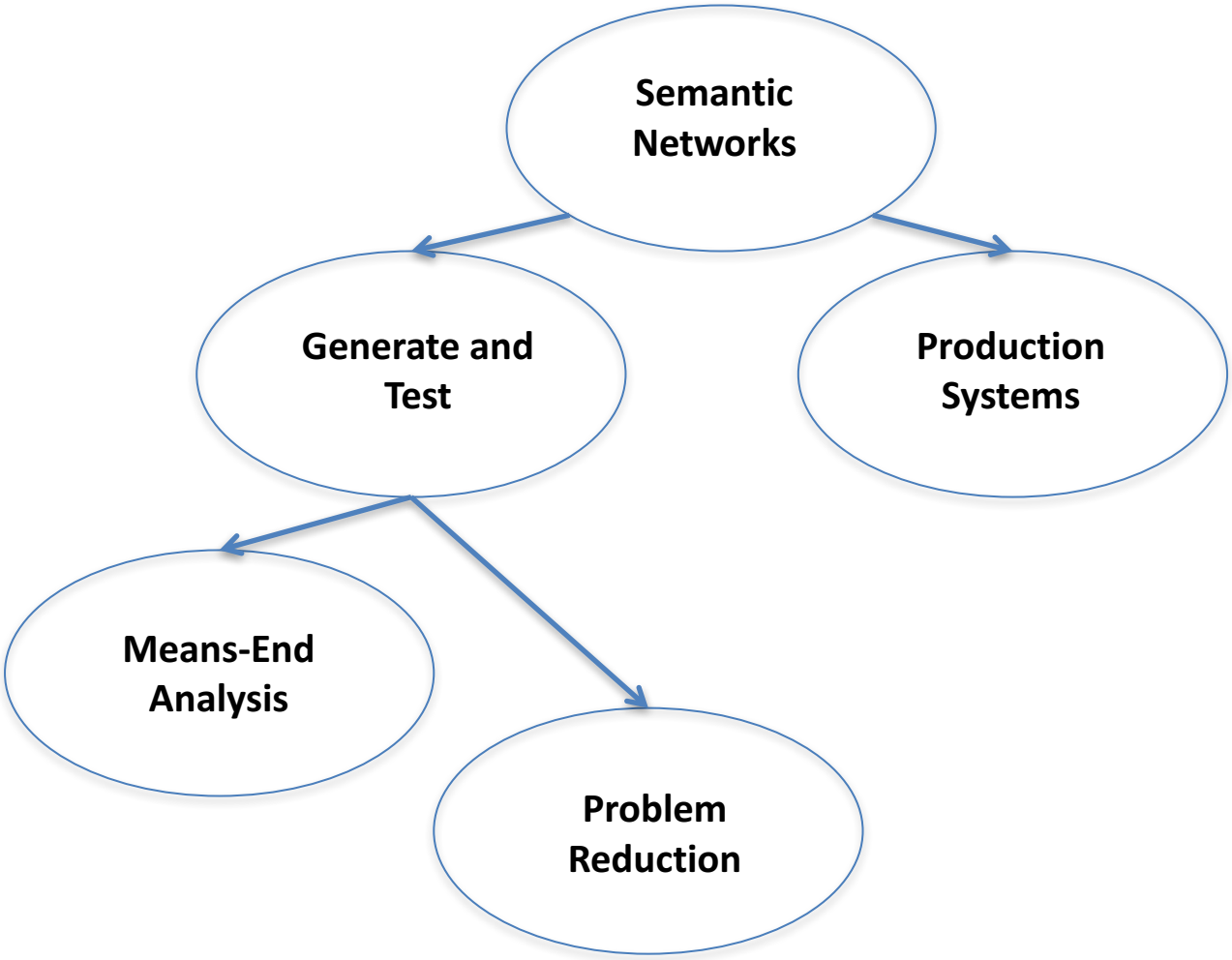
Cognitive Computing and Systems Seminar

Knowledge-Based AI: Cognitive Systems Fundamentals

Topics

Fundamentals	Planning	Common Sense Reasoning
Learning	Knowledge-Based Artificial Intelligence	Analogical Reasoning
Visuospatial Reasoning	Design Creativity	Meta Cognition

Fundamentals



Semantic Networks

- Semantics networks as "knowledge representation"
  - (1) language/expression - vocabulary
  - (2) content
- Example
  - f=ma
  - (1) algebraic equation
  - (2) content is each of force = mass * acceleration

Semantic Networks

- Semantic Network Structure:
  lexicon : nodes/object
  structure: directional links
  semantics: application specific labels(like transformation type e.g. deleted, above, inside, etc)

- GOOD representations ==> transparent, concise, complete, fast, computable
  makes relationships explicit
  exclude extraneous details
  expose natural constrains
  brings objects and relations together

- NOTE: good representation helps problem solving; but, no representation method is perfect, each representation has its own affordance and constraints (Guards and Prisoners Problem)

Fundamentals

Semantic Networks

Semantic Networks

How to represent *Raven's Progressive Matrices* using a semantic network. State A, and state B.

- Label all objects (x is a circle, y is the diamond, z is the black dot), and reference them as nodes
- Represent the relationships between nodes, in both states (frames), both A and B.
- Represent the transformation between the nodes between states, A and B.

Structure of Semantic Networks

- Lexically: nodes
- Structurally: directional links
- Semantically: application-specific labels

Characteristics of Good Representations

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

Fundamentals

Semantic Networks

Guards and Prisoners Problem

Description

- Three guard and three prisoners must cross river.
- Boat may take only one or two people at a time.
- Prisoners may never outnumber guards on either time (thought prisoner may be alone on either coast).

Modeling using Semantic Network

Lexicon: Consider each node to be a unique state, represented by: - number of prisoners and guards on left side - number of prisoners and guards on right side - side that boat is on.

Inference about State Transitions?

Which transitions (e.g. moves) between states are both legal AND productive? Represent total possible states given transformations possible:

i	0	1	2	3	4	6	7	8	9	10	11	12
trans :	init: <	2p to R:>	p to L:<	2p to R:>	p to L:<	2g to R:>	g.p to L:<	2g to R:>	p to L:<	2p to R:>	p to L:<	2p to R:<
g_i	3, 0	3, 0	3, 0	3, 0	3, 0	1, 2	2, 1	0, 3	0, 3	0, 3	0, 3	0, 3
p_i	3, 0	1, 2	2, 1	0, 3	1, 2	1, 2	2, 1	2, 1	3, 0	1, 2	2, 0	0, 3

Knowledge Representation as a medium for human expression

- An intelligent system must have knowledge representations (KR) that can be interpreted by humans.
  - We need to be able to encode information in the knowledge base without significant effort.
  - We need to be able to understand what the system knows and how it draws its conclusions.

Knowledge Representation

Knowledge Representation

- Logic (prepositional, predicate)
- Network representation
  - Semantic nets
- Structured representation
  - Frames
- Issues in KR
  - Hierarchies, inheritance, exceptions
- Advantages and disadvantages

Knowledge Representation

Semantic Networks

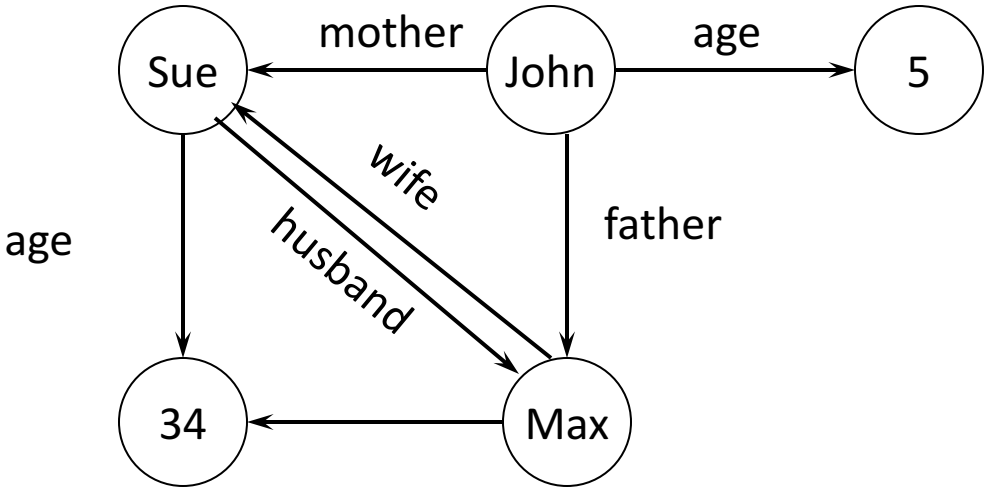
- First introduced by Quillian back in the late-60s

  M. Ross Quillian. "Semantic Memories", In M. M. Minsky, editor, *Semantic Information Processing,* pages 216-270. Cambridge, MA: MIT Press, 1968

- **Semantic network** is simple representation scheme which uses a graph of labeled nodes and labeled directed arcs to encode knowledge
  - Nodes – objects, concepts, events
  - Arcs – relationships between nodes
- **Graphical depiction** associated with semantic networks is a big reason for their popularity

Nodes and Arcs

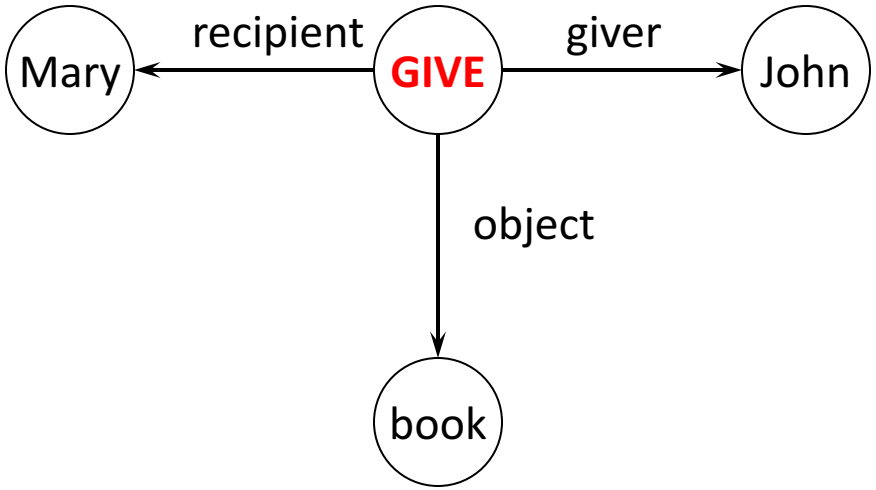
- Arcs define binary relations which hold between objects denoted by the nodes.



mother (john, sue)
age (john, 5)
wife (sue, max)
age (max, 34)
…

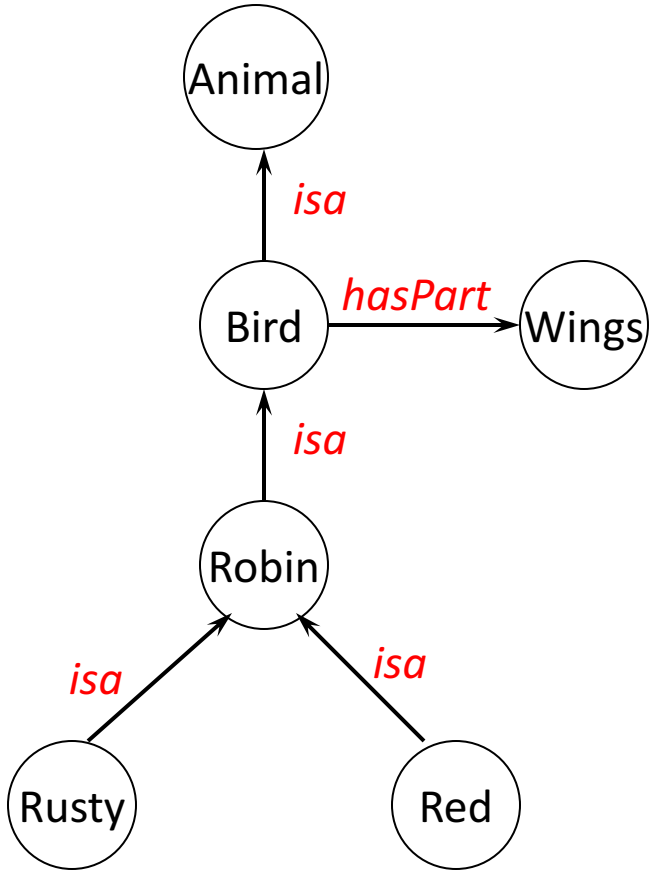
Non-binary relations

- We can represent the generic *give* event as a relation involving three things:
  - A giver
  - A recipient
  - An object



Inheritance

- Inheritance is one of the main kind of reasoning done in semantic nets
- The **ISA** (is a) relation is often used to link a class and its superclass.
- Some links (e.g. **haspart**) are inherited along **ISA** paths
- The semantics of a semantic net can be relatively informal or very formal
  - Often defined at the implementation level

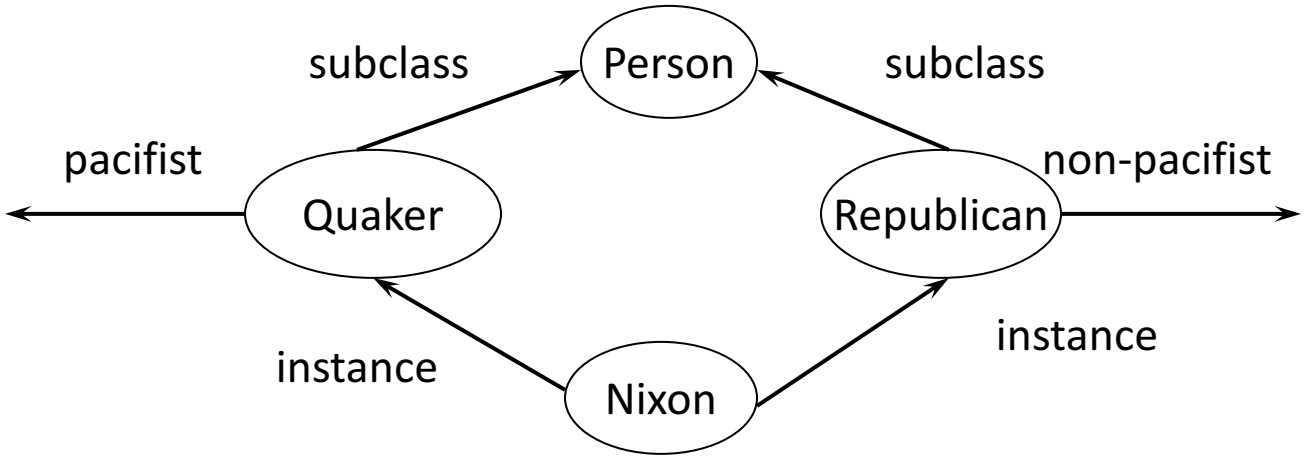


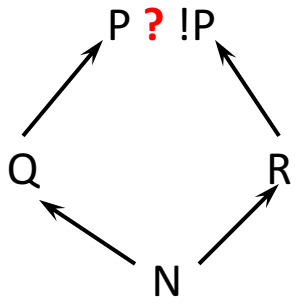
Multiple Inheritance

- A node can have any number of superclasses that contain it, enabling a node to inherit properties from multiple *parent* nodes and their ancestors in the network. It can cause conflicting inheritance.

Nixon Diamond

(two contradictory inferences from the same data)





Advantages of Semantic nets

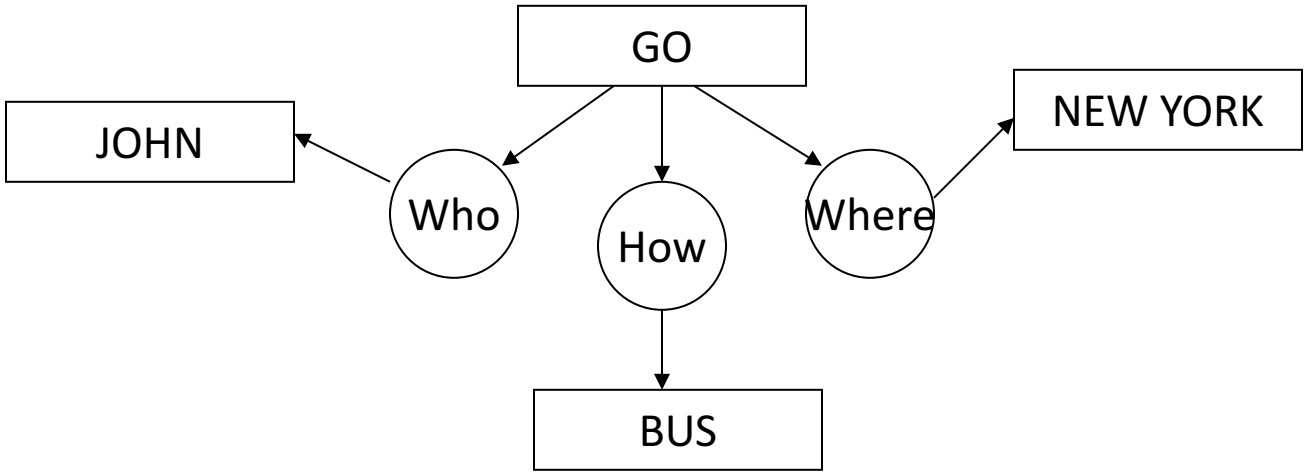
- Easy to visualize
- Formal definitions of semantic networks have been developed.
- Related knowledge is easily clustered.
- Efficient in space requirements
  - Objects represented only once
  - Relationships handled by pointers

Disadvantages of Semantic nets

- Inheritance (particularly from multiple sources and when exceptions in inheritance are wanted) can cause problems.
- Facts placed inappropriately cause problems.
- No standards about node and arc values

Conceptual Graphs

- *Conceptual graphs* are semantic nets representing the meaning of (simple) sentences in natural language
- Two types of nodes:
  - *Concept nodes*; there are two types of concepts, individual concepts and generic concepts
  - *Relation nodes*(binary relations between concepts)



Frames

- Frames – semantic net with properties
- A frame represents an entity as a set of slots (attributes) and associated values
- A frame can represent a specific entry, or a general concept
- Frames are implicitly associated with one another because the value of a slot can be another frame

3 components of a frame

- frame name
- attributes (slots)
- values (fillers: list of values, range, string, etc.)

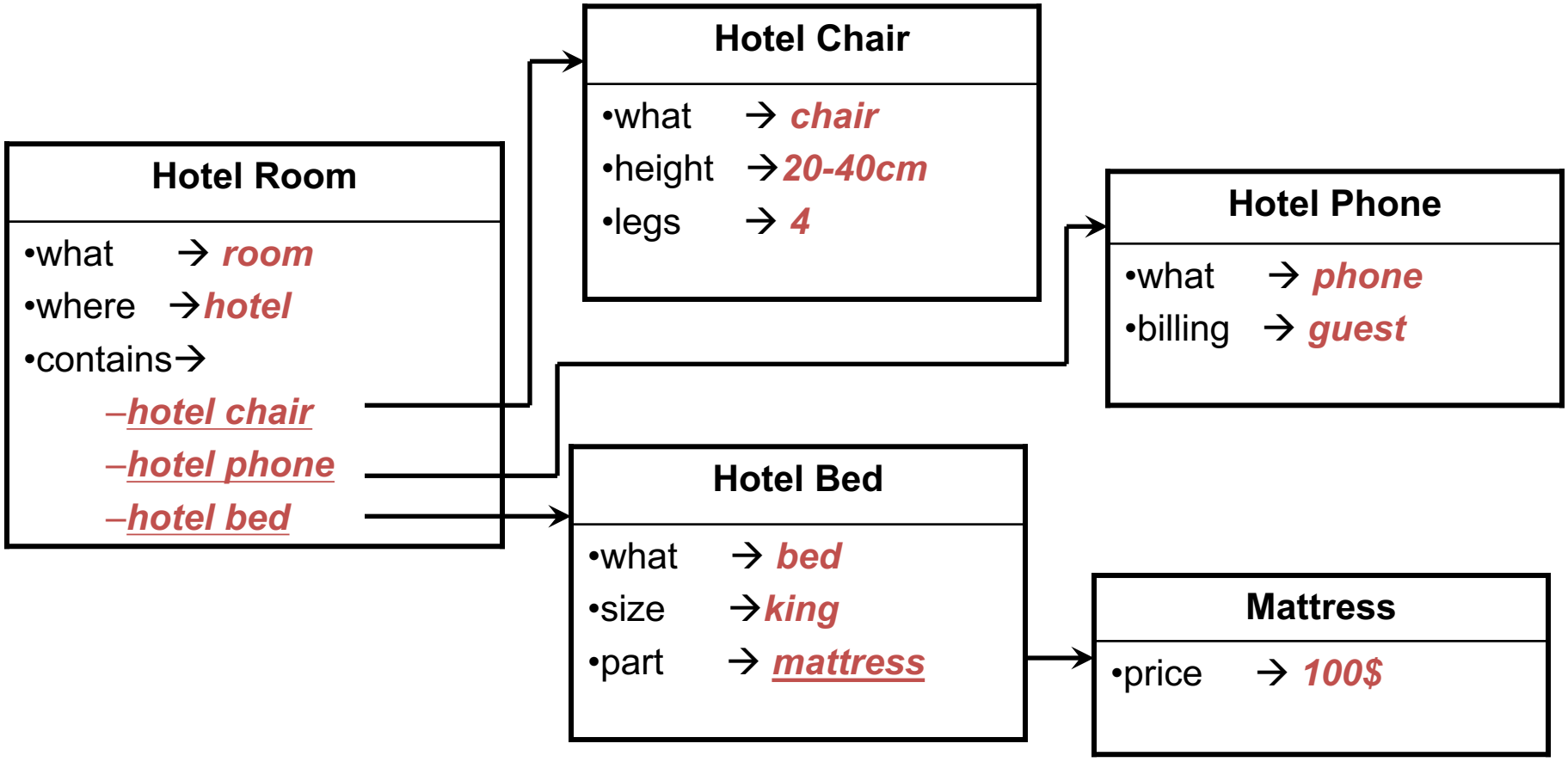
Book Frame	
Slot →	*Filler*
Title →	*AI. A modern Approach*
Author →	*Russell & Norvig*
Year →	*2003*

Features of Frame Representation

- More natural support of values then semantic nets (each slots has constraints describing legal values that a slot can take)
- Can be easily implemented using object-oriented programming techniques
- Inheritance is easily controlled

Inheritance

- Similar to Object-Oriented programming paradigm



Benefits of Frames

- Makes programming easier by grouping related knowledge
- Easily understood by non-developers
- Expressive power
- Easy to set up slots for new properties and relations
- Easy to include default information and detect missing values

Drawbacks of Frames

- No standards (slot-filler values)
- More of a general methodology than a specific representation:
  - Frame for a class-room will be different for a professor and for a maintenance worker
- No associated reasoning/inference mechanisms

Generate and Test

- Problem Solving Method:
  - Example: guards and prisoners generate all possible states, and test them.
  - (test and prune illegal, unproductive ones) - smart/dumb generator/tester efficiency (imagine millions of possible states, then obviously generator should be smart).

Generate and Test

Choosing Matches by Weight

One can weigh transformations to favor specific types of transformations over others. For example:

Points Weights

- 5: Unchanged
- 4: Reflected
- 3: Rotated
- 2: Scaled
- 1: Deleted
- 0: Shape Changed

Generators and Testers

Smart Generator

Takes one state, and generates all possible states, however, should not generate states that are not productive.

Smart Tester

Analyzes all generated states, and filters to only legal (i.e., those states that adhere to logical constraints) and productive (i.e., resultant states that have not yet been observed) states.

Note: For each problem, we have to find the right balance the smartness of generators and testers.

Generate and Test

Generate and Test for Raven's Problems

Explicit

1. Setup semantic network for A and B, and construct transformation.
2. Apply transformation to C to generate D
3. Compare D to possible solutions and find highest match (via level of confidence).
4. Ensure that D meets a confidence level threshold.
5. If not, repeat 1-4, however using new semantic network construction approach.

Implicit

1. Setup semantic network for A and B and construct transformation
2. Take a solution (1-6) and C and setup semantic network and construct transformation.
3. Compare resultant transformation
4. Repeat for every solution to find best match (via weighting)

Means-End Analysis

- Means-End Analysis:
  - compare the current state and goal state
  - find the difference (aka distance/cost) between "new" state and goal state
  - pick an operator that minimizes diff between them
- Means-End Analysis is useful when we know enough information to be able to define the distance evaluation function correctly.

Means-End Analysis

Means-end Analysis and Problem Reduction
Useful for *well-formed* problems.

State Spaces
The Block Problem

- You may only move one block at a time.
- You may only move blocks that have nothing on top of them.

Move(C, Table)
Move(B, C)
Move(A, B)

Definition
Have some sort of *Initial State* to a *Goal State*. From state transitions are performed via valid operations.

Means-End Analysis

Means-End Analysis

For each operator that can be applied:

- Apply the current operator to the current state.
- Calculate the difference between the current state and the goal state.
- Prefer the state the minimizes the distance between the new state and goal state.

Note: as a general method, means-end can be applied to a varied range of problems. Due to it generality, it can however, get caught in local minima and cannot converge to the global solution.

Problem Reduction

Take the goal of a problem and break it apart into more simple goals. Apply means-end after.

Problem Reduction

- Problem Reduction:
  - divide and conquer.
  - not necessarily "overlapping sub-problems and optimal substructure" but at least sub-problems.
- Note: The link between Semantic Network (knowledge representation) and PR, G&T, MEA (problem solving method) are "weak" because they make little use of knowledge.

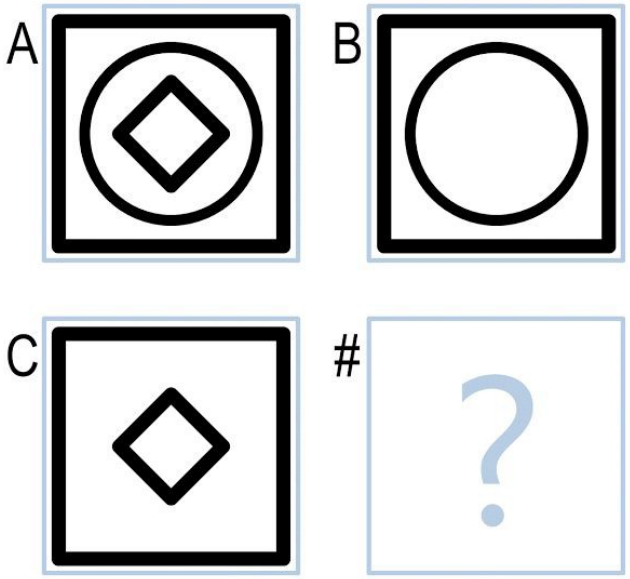
Production Systems

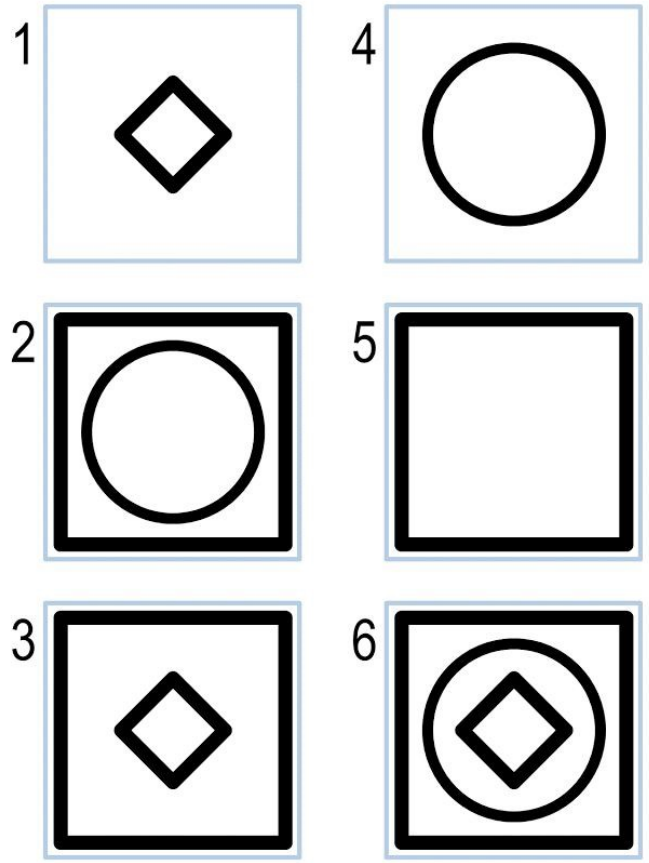
- Production Systems relate to "learning"
  - Production Systems use a cognitive architecture in which knowledge is represented as "rules"
  - Production Systems employ a particular mechanism of learning called "chunking"

- Note: Baseball Example

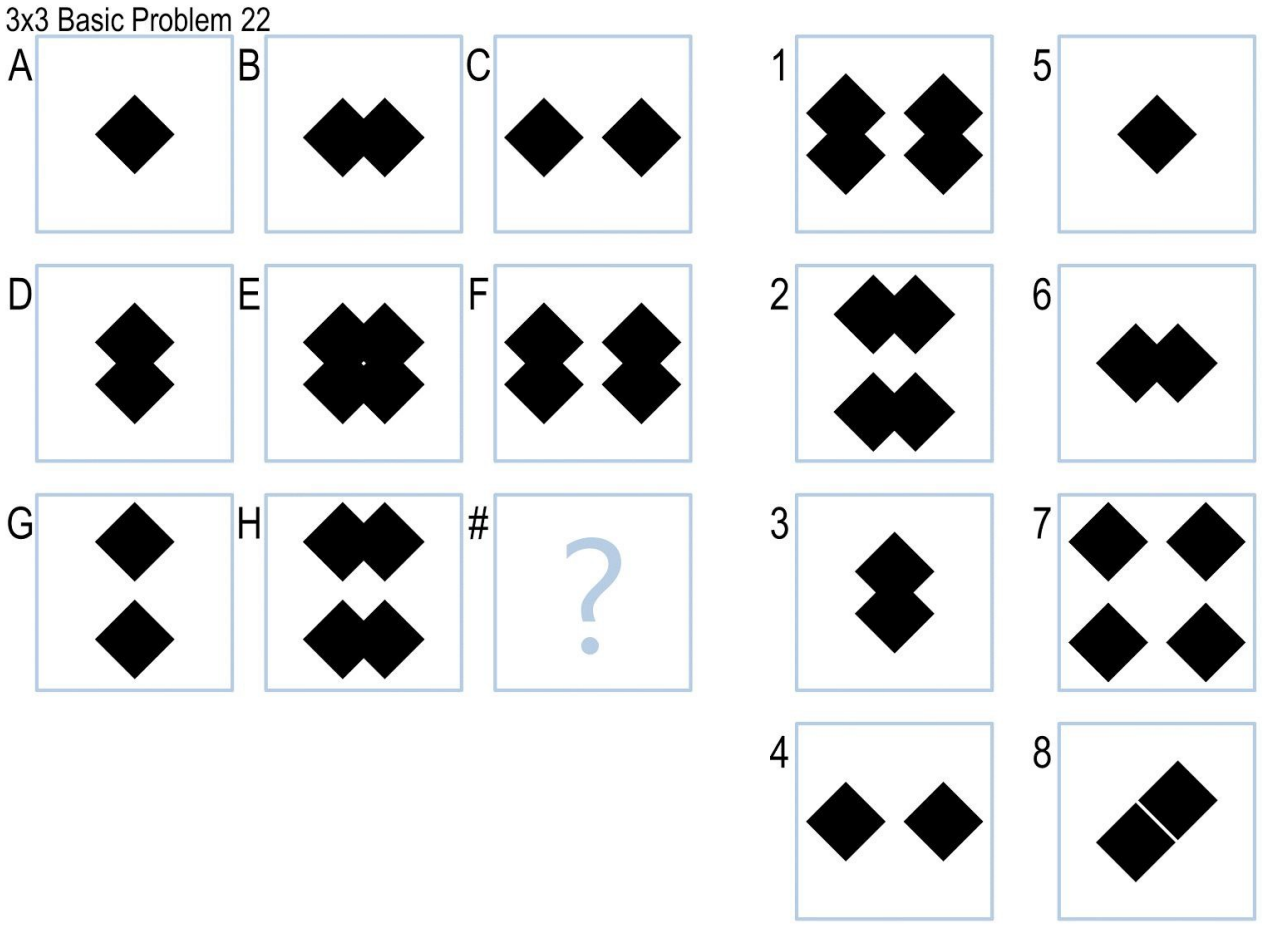
Raven's Progressive Matrices

2x2 Basic Problem 12

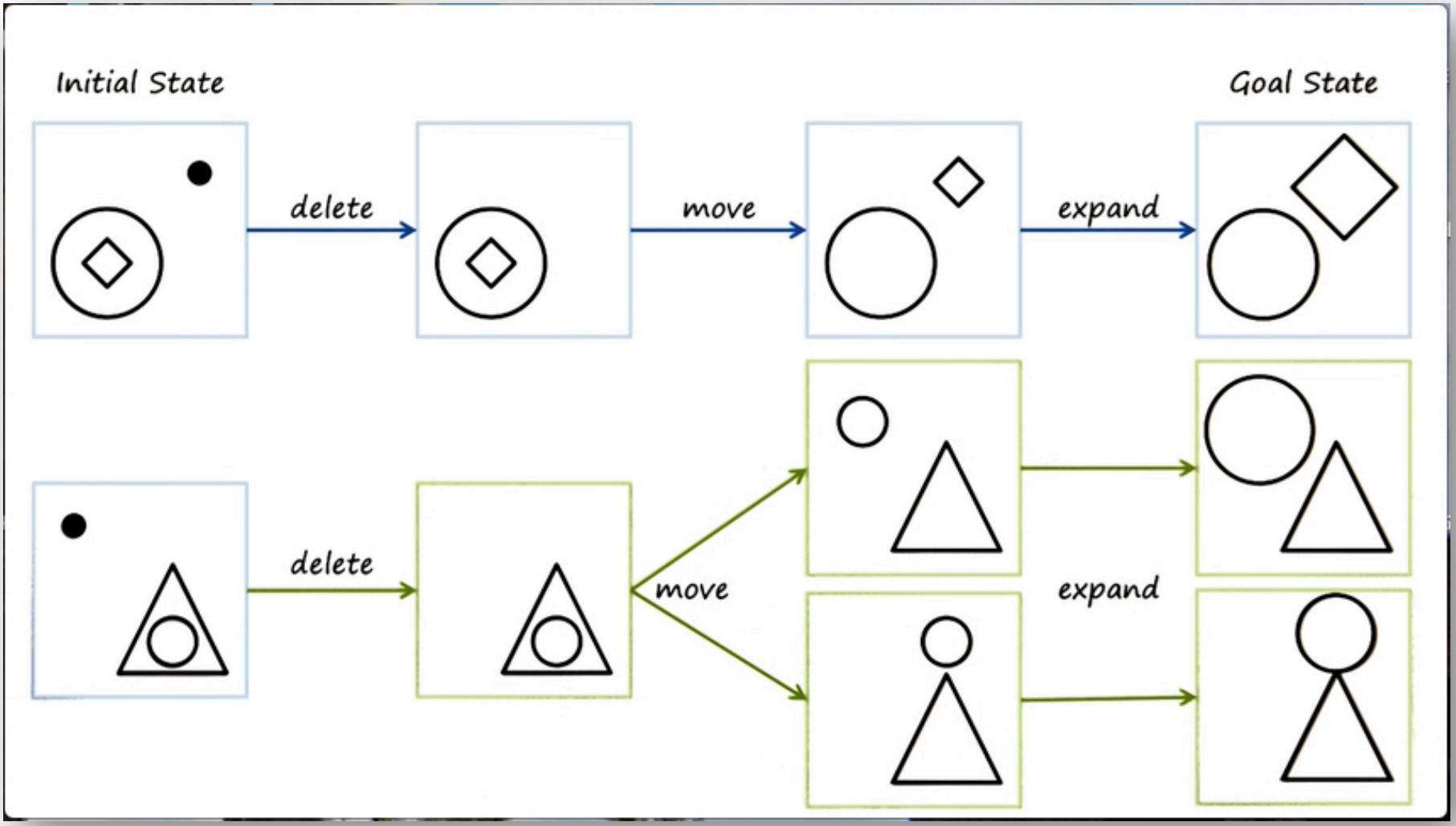




Raven's Progressive Matrices



Means-End Analysis



RAVENS

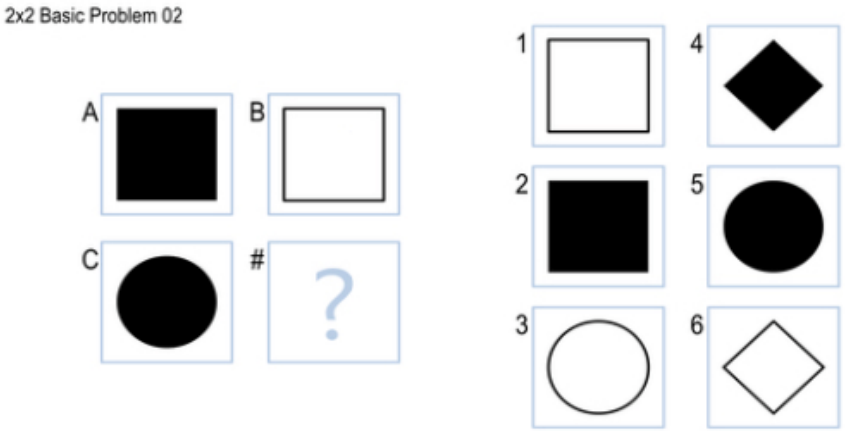
Means-End Analysis

Means-End Analysis

- In terms of Means-End analysis, the Agent would first evaluate the goal state. In order to determine what the goal state should be, it will evaluate the pairings of AB and AC, and dump the knowledge differences into a storage space.

  AB Space = {“fill”}
  AC Space = {“shape”} Problem2Space = {“fill”, “shape”}

- After we have some inference about our goal, our agent can then loop through each answer choice, and eliminate answers that don’t bring our agent closer to the goal.



RAVENS

Means-End Analysis

Pseudo Code

```
For currFigure in Problem.getFigures():
    AFigSpace = analyze(A, currFigure)
    CFigSpace = analyze(C, currFigure)
    totalSpace = comine(AFigSpace, CFigSpace)
    for element in Problem2Space:
     if(totalSpace does not contain(element))
          eliminate(currFigure)
```

So through this simple pseudo code, our agent is eliminating the possible answer choices that contain differences that have no correlation to the goal state. This is exactly what Means-End analysis provides, steps that only lead closer to the goal.

RAVENS

Means-End Analysis

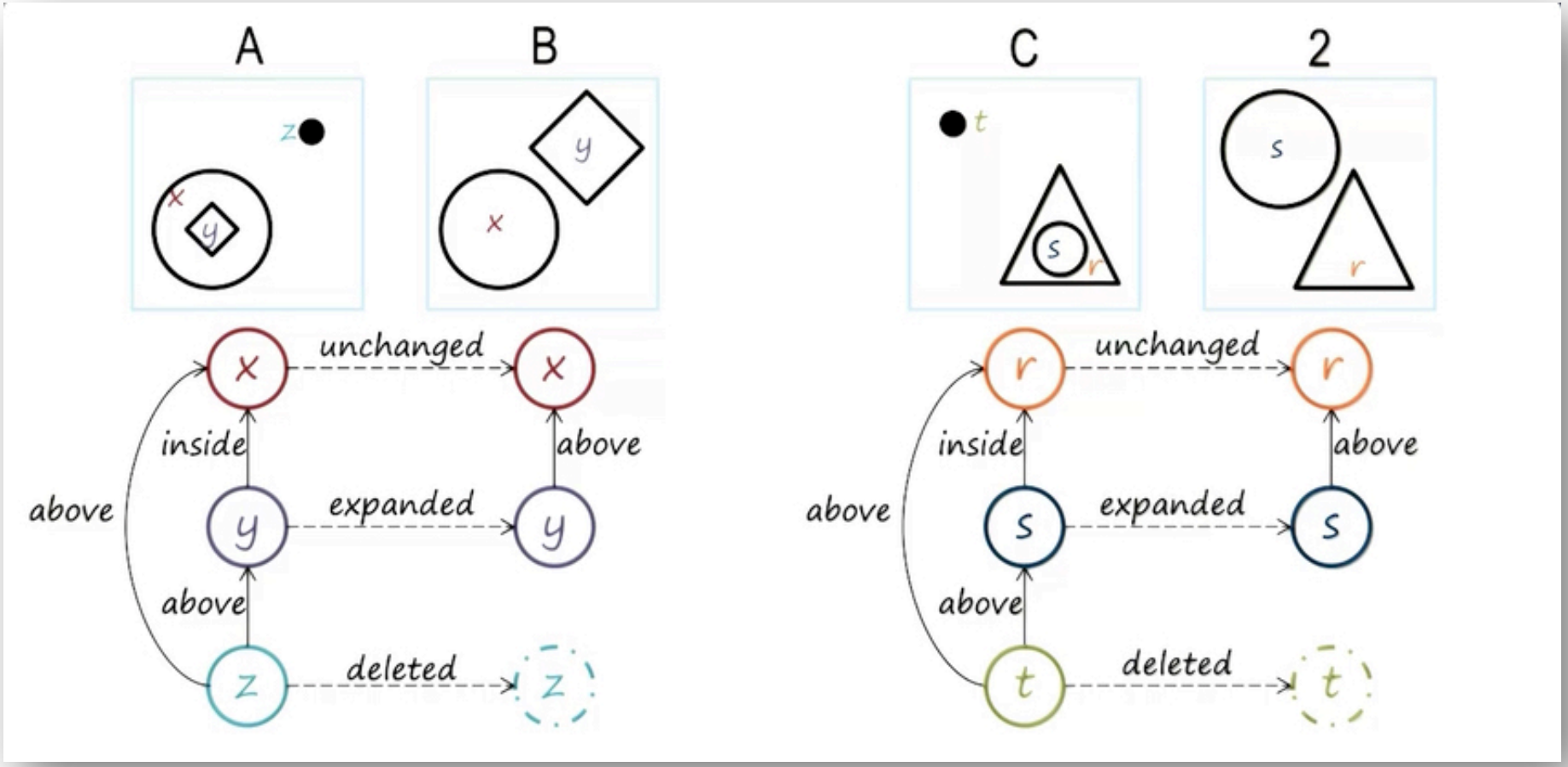
Means-End Analysis

So if we applied this pseudo code to the 2x2 Problem 2:

- Figure 1 – would be kept as possible answer choice because it's space would be the exact same Problem2Space - shape and fill differences.
- Figure 2 – this figure would be eliminated because it does not contain a fill difference.
- Figure 3 – would be kept because it follows the same procedures to become closer to the goal state
- Figure 4 – would be eliminated because it contains an unnecessary trait of angle change, which does not bring us any closer to the goal state. It also has no fill change.
- Figure 5 – Would be eliminated as well, no fill change, which is necessary to reach goal state.
- Figure 6 – Unnecessary angle change, does not help our agent reach the goal state

RAVENS

Semantic Networks



RAVENS

Semantic Networks

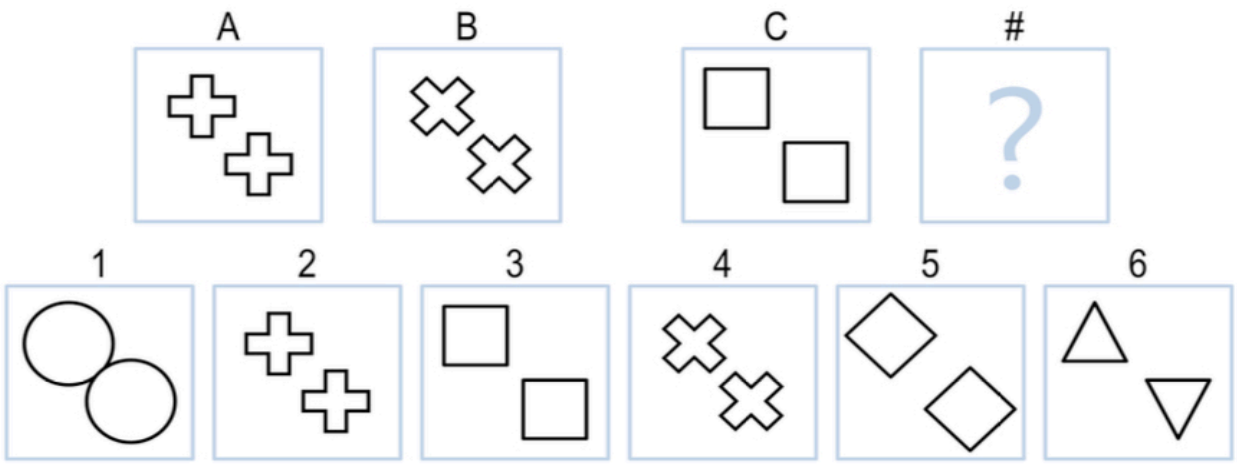
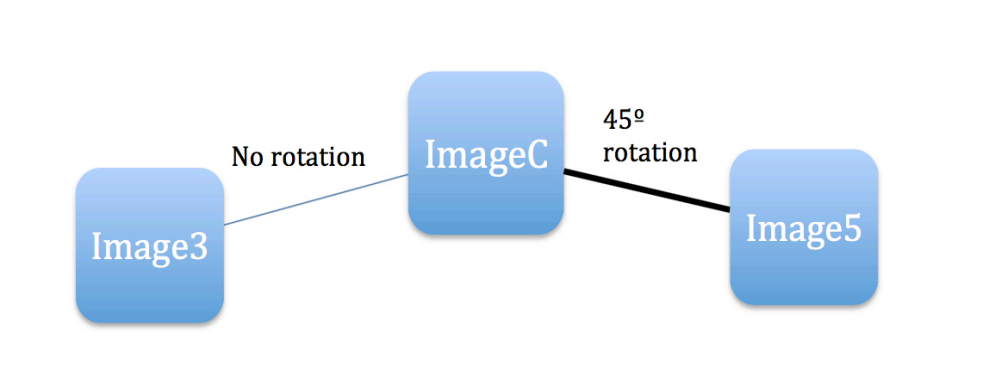


Image A = { Object1(Plus, Upper-Left, 30x30), Object2(Plus, Lower-Right, 30x30) }

Image B = { Object1(Plus, Upper-Left, 30x30), Object2(Plus, Lower-Right, 30x30) }

RAVENS

Semantic Networks

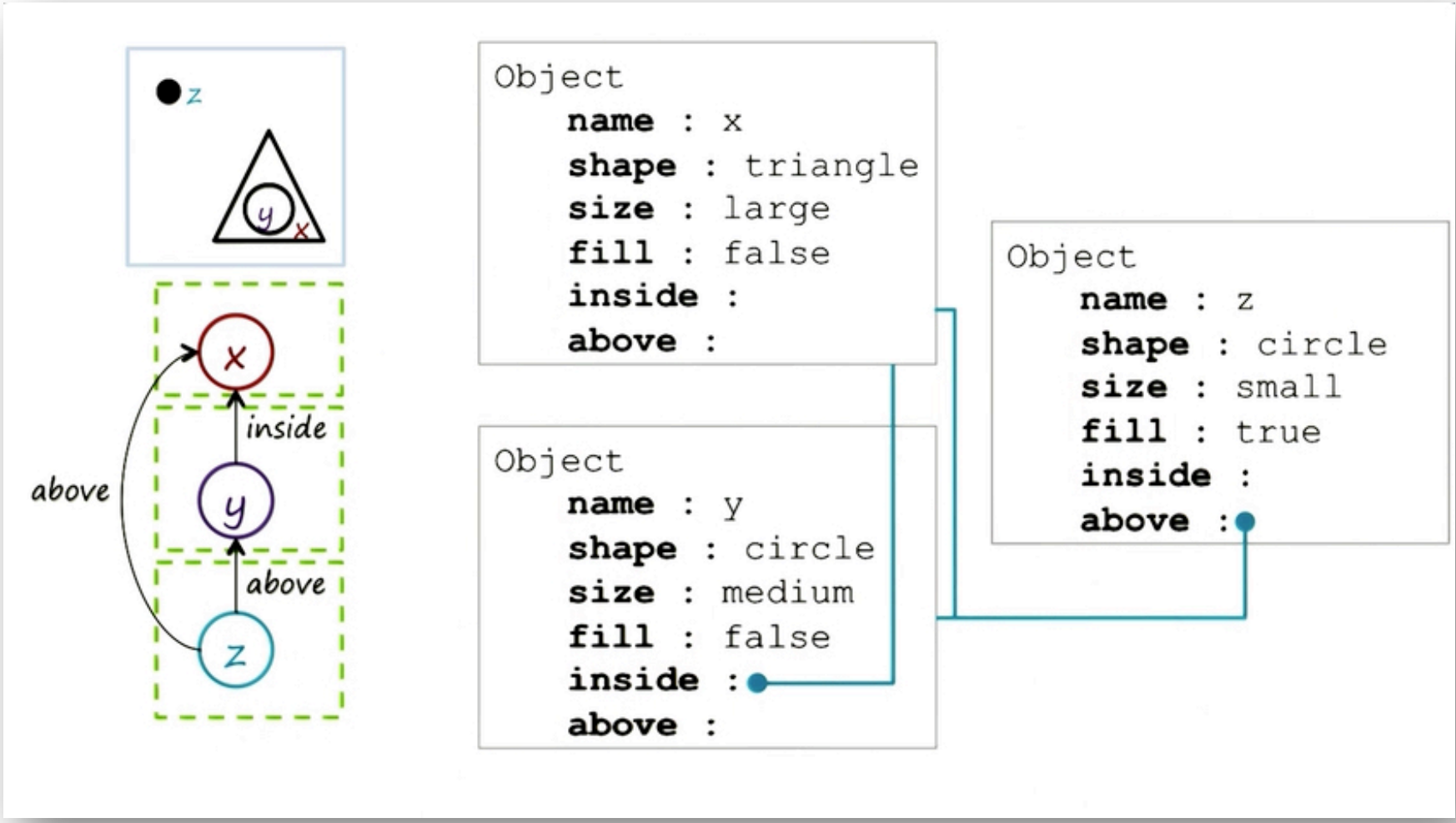


For this situation, the agent would notice that the characteristics are the same, however the pixels are slightly different – for in fact the only change between image A and B is that the two objects were rotated 45 degrees to the right.

An Analogy function is run at the end as a last-case scenario because many images can be eliminated without it. For example, because A and B both have the same two objects in each image, we know for C that images 1,2,4, and 6 could never be it's match – they are immediately eliminated!

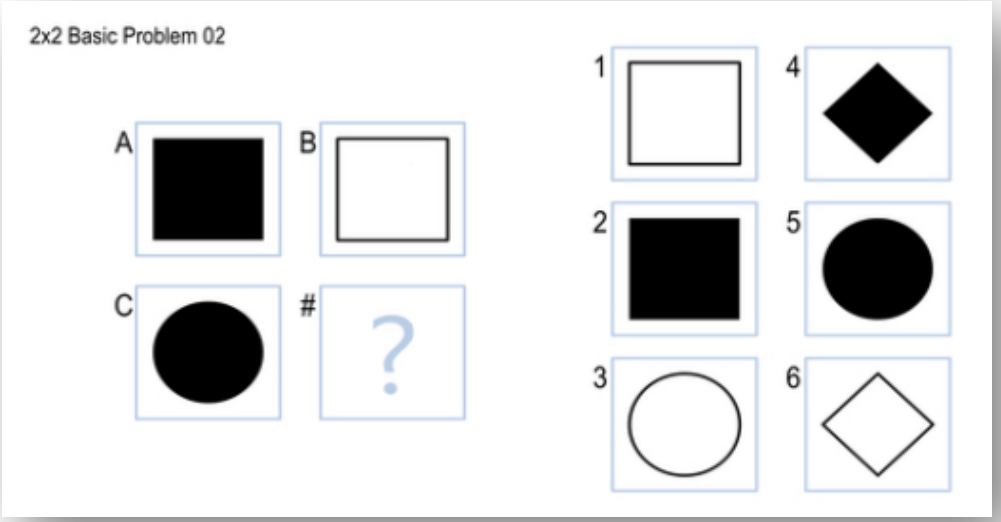
RAVENS

Frames



RAVENS

Frames



2x2 Problem Frame:

{

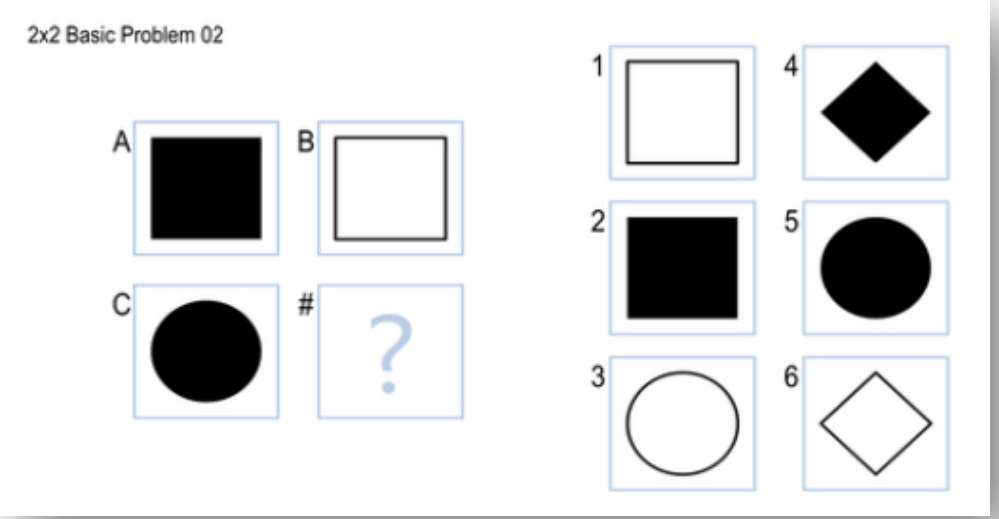
Type: basic problem
A: figureA)
B: figure(B)
C: figure(C)
NumShapesEqual: true
AmountShapesAnswerChoices: 1
Answer: 5
ABDiff: 1

}

RAVENS

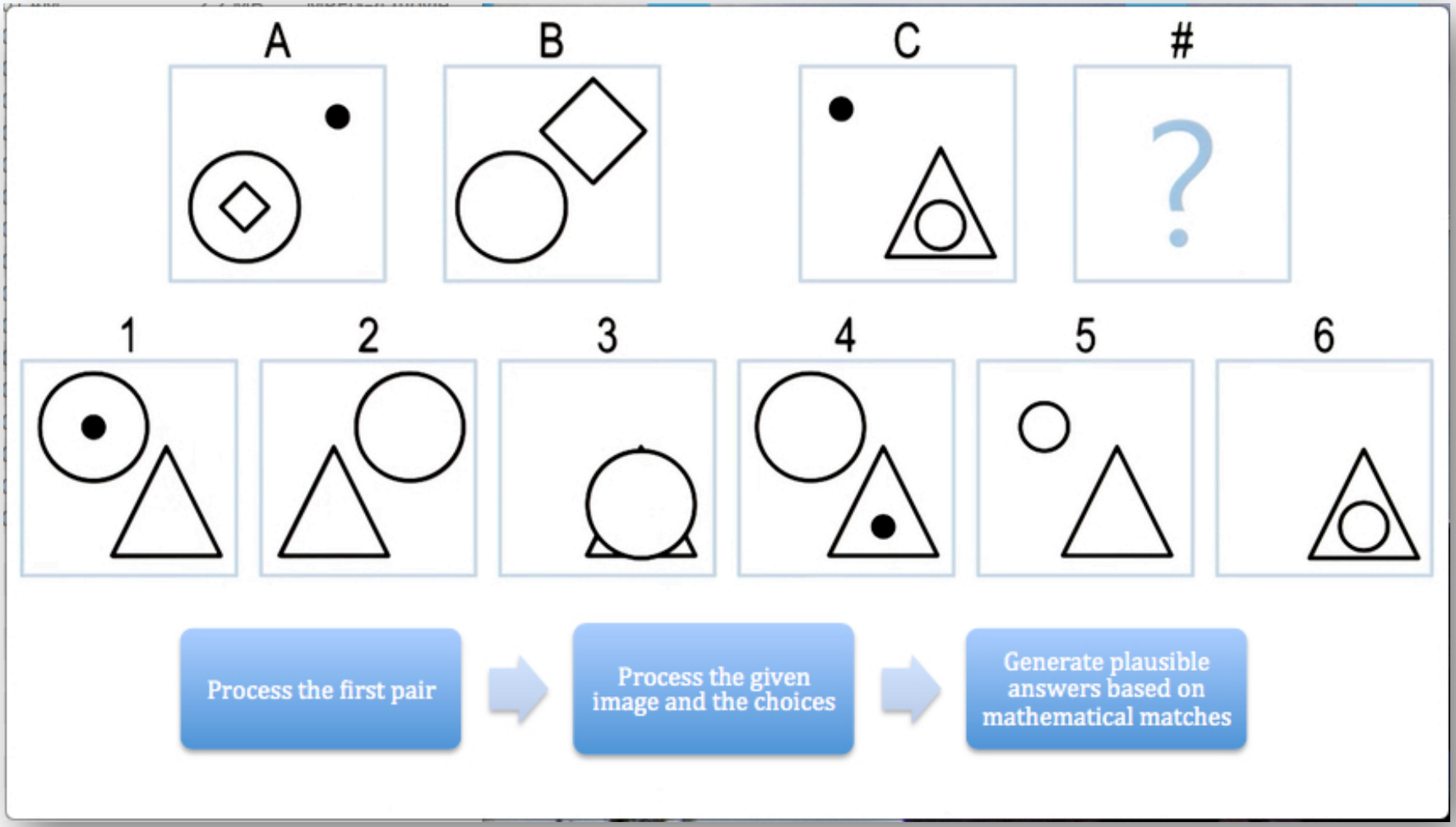
Frames

2x2 Object Frame:
{
    name: A
    shape: square
    fill: yes
    size: large
    angle: null
}



RAVENS

Generate and Test



RAVENS

Generate and Test

Pseudo Code:

process(image) – takes in an image and then mathematically gives it a rating based on it's objects size, type, and location and returns the stats in a wrapper object.

compare(imageAStats, imageBStats) – takes in two image wrapper objects with their ratings and compares the difference.

ravensProgressiveMatrices(imageMatch1, imageMatch2, imageA, totalChoices) { //Sections resemble breaking down the problem into smaller steps

image1Object = process(imageMatch1); image2Object = process(imageMatch2); imageAObject = process(imageA)

stats = compare(image1Object, image2Object)

for image i in totalChoices:

    imageCurrObject= process(i)
    currStats = compare(imageAObject, imageCurrObject) abs(diff) = stats – currStats
    store diff and object in max heap or list

imageAnswer = pop off max heap/structure

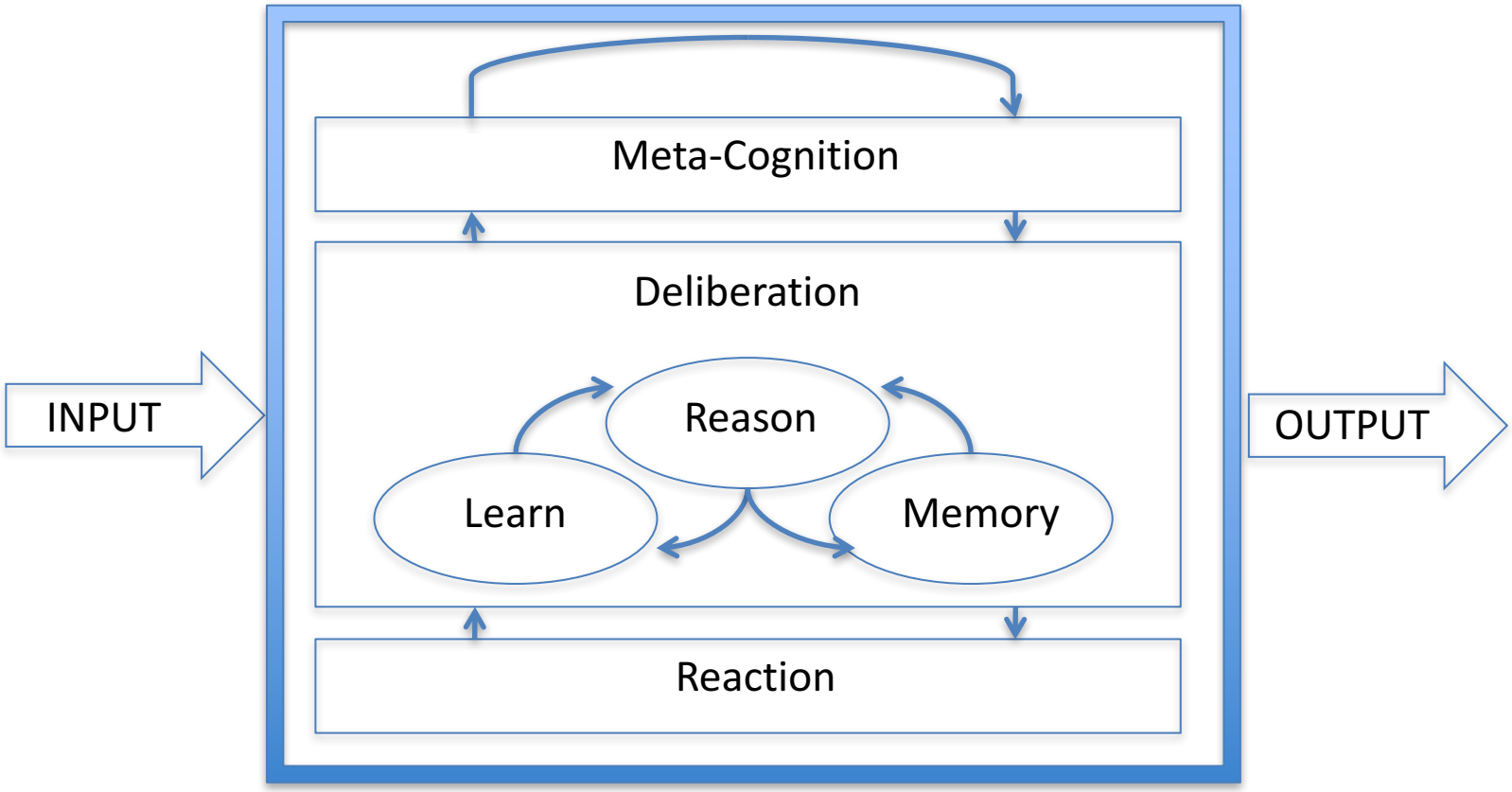
return imageAnswer }

RAVENS

Principles

1. KBAI Agents Represent and Organize Knowledge into Knowledge Structures to Guide and Support Reasoning
2. Learning in KBAI Agents is often Incremental
3. Reasoning in KBAI Agents is Top-Down as well as Bottom-Up
4. KBAI Agents Match Methods to Tasks
5. KBAI Agents use Heuristics to Find Solutions that are God Enough and Not Necessarily Optimal
6. KBAI Agents make use of Recurring Patterns in the Problems They Solve
7. The Architecture of BAI Agents Enables Reasoning, Learning, and Memory to Support and Constrain Each Other

Cognitive Systems Structure



Meta-Cognition