1. Intro to Knowledge Based AI

1.1 Fundamental conundrums of AI:
- 1.1.1 Intelligent agents have limited resources.
- 1.1.2 Computation is local, but problems have global constraints.
- 1.1.3 Logic is deductive, but many problems are not.
- 1.1.4 The world is dynamic, but knowledge is limited.
- 1.1.5 Problem solving, reasoning, and learning are complex, but explanation and justification are even more complex.

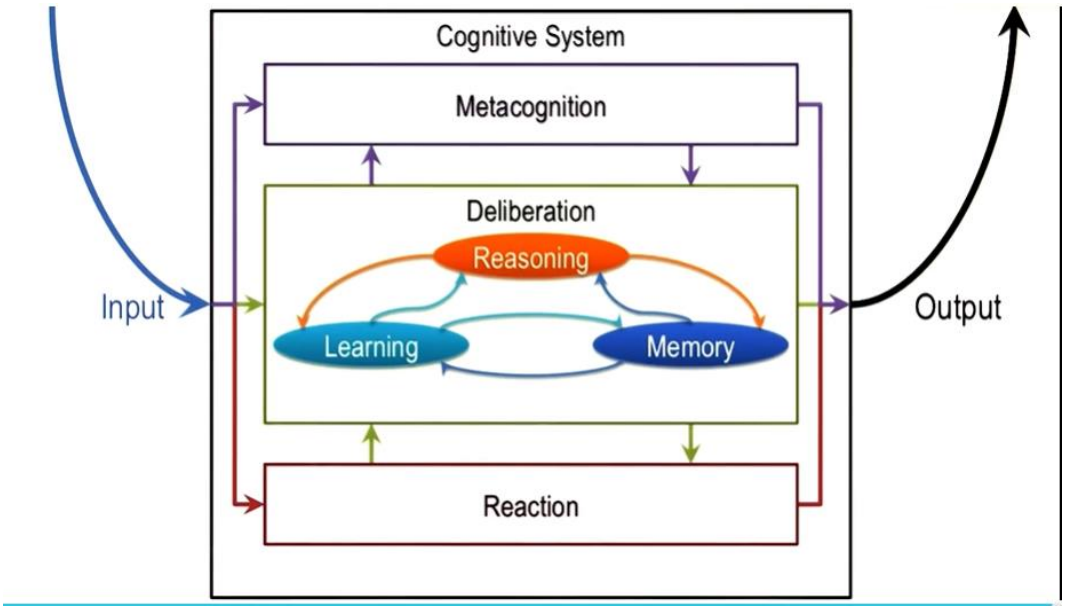
1.2 Characteristics of AI problems:
- 1.2.1 Knowledge often arrives incrementally.
- 1.2.2 Problem exhibit recurring patterns.
- 1.2.3 Problems have multiple levels of granularity.
- 1.2.4 Any problems are computationally intractable.
- 1.2.5 The world is dynamic, but knowledge of the world is static.
- 1.2.6 The world is open-ended, but knowledge is limited.

1.3 Characteristics of AI agents:
- 1.3.1 Agents have limited computing power.
- 1.3.2 Agent have limited sensors.
- 1.3.3 Agents have limited attention.
- 1.3.4 Computational logic is fundamentally deductive.
- 1.3.5 AI agents' knowledge is incomplete relative to the world.

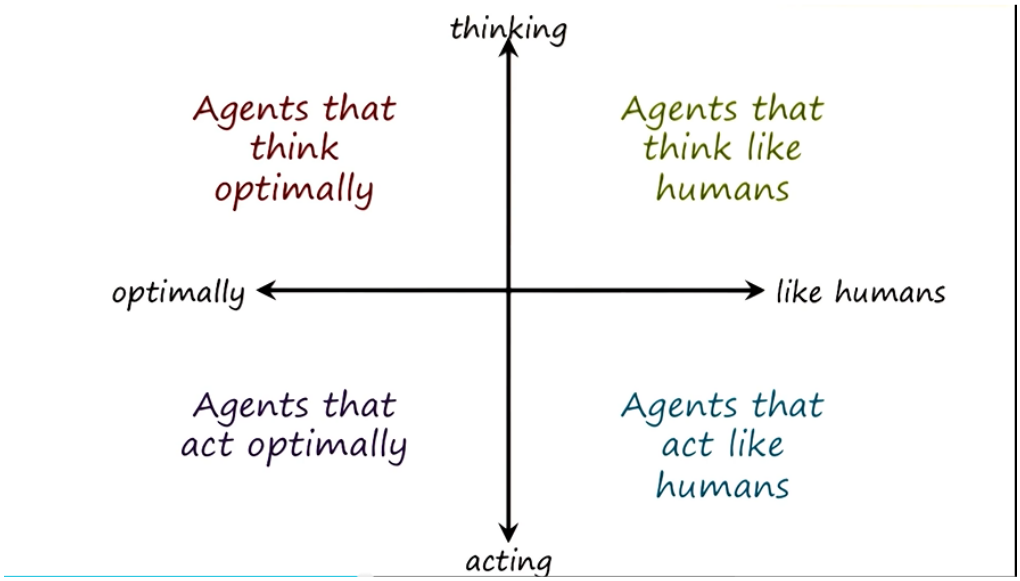
1.4 Example – to play Jeopardy, Watson must:
- 1.4.1 read the clue,
- 1.4.2 search through its knowledge base,
- 1.4.3 decide on its answer,
- 1.4.4 phrase the answer in form of the question.

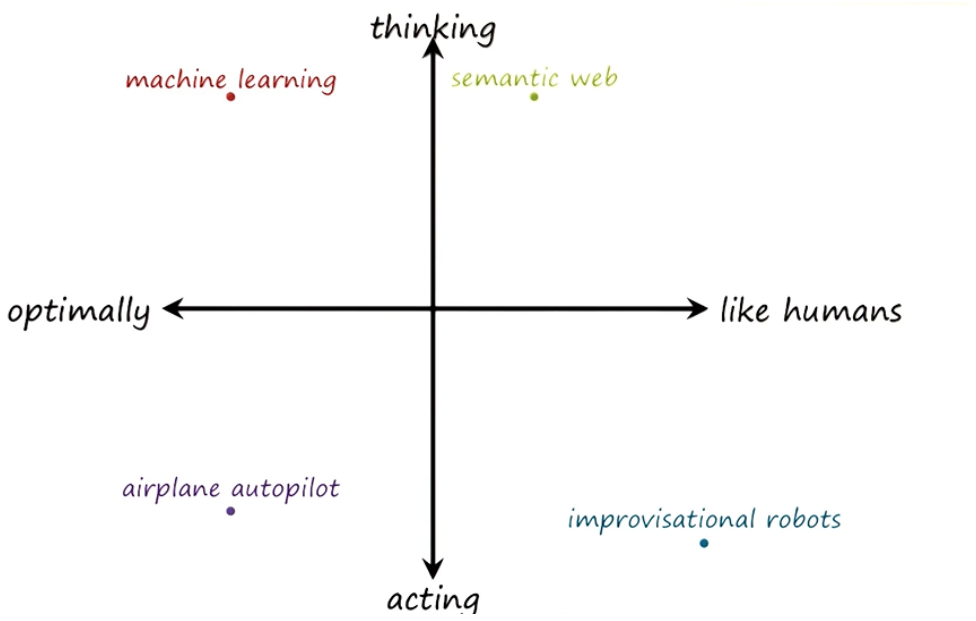
1.5 Fundamental processes of KBAI – deliberation:
- 1.5.1 Reasoning
- 1.5.2 Learning
- 1.5.3 Memory

1.6 Overall architecture of AI:

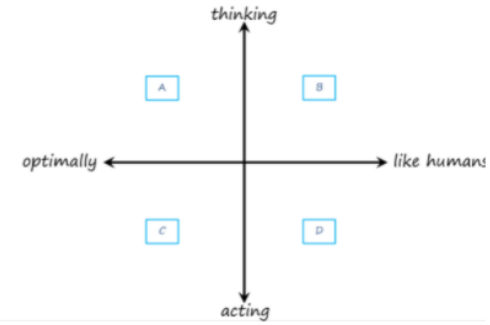


1.7 The four schools of AI:





A – Google Maps (route navigation)
B – Siri (Apple virtual assistant)
C – Roomba (automated vacuum)
D – C-3PO (Star Wars)



1.8 What are cognitive systems?

1.8.1 Cognitive – dealing with human-like intelligence

1.8.2 Systems – multiple interacting components such as learning, reasoning and memory

1.8.3 Cognitive systems – systems that exhibit human-like intelligence through processes like learning, reasoning, and memory

1.9 Topics in KBAI:

1.9.1 Fundamentals:

Semantic networks

Generate and test

Means-end analysis

Problem reduction

Production Systems

1.9.2 Planning:

Logic

Planning

1.9.3 Common sense reasoning:

Frames

Understanding
Common sense reasoning
Scripts

1.9.4 Analogical reasoning:
Learning by recording cases
Case-based reasoning
Explanation-based reasoning
Analogical reasoning

1.9.5 Metacognition:
Learning by correcting mistakes
Meta-reasoning
Ethics in AI

1.9.6 Design & creativity:
Configuration
Diagnosis
Design
Creativity

1.9.7 Visuospatial reasoning:
Constraint propagation
Visuospatial reasoning

1.9.8 Learning:
Learning by recording cases
Incremental concept learning
Classification
Version spaces

2. Introduction to CS7637

2.1 Computational psychometrics is a design of computational agents that can take the same kind of test that humans do when tested for intelligence or knowledge

2.2 Raven's Progressive Matrices:

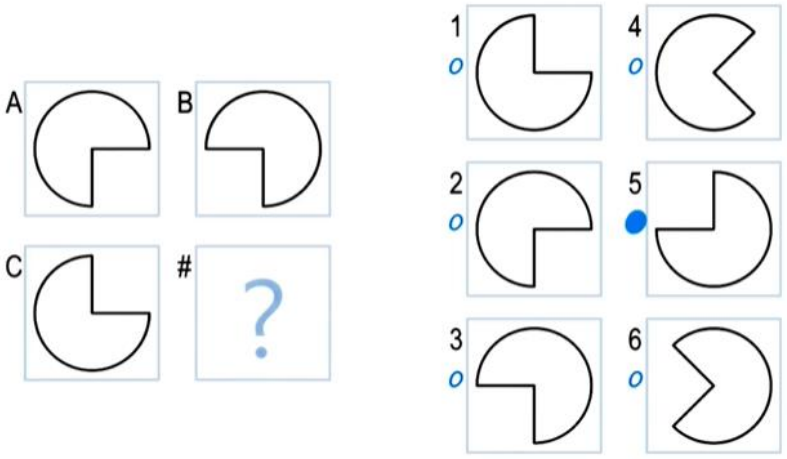
- 2.2.1 Test written in the 1930s to examine general intelligence
- 2.2.2 60 multiple-choice visual analogy problems
- 2.2.3 Unique, as all problems are strictly visual
- 2.2.4 A widespread test
- 2.2.5 2x2 (vertical, horizontal, diagonal relationship), 3x3 (vertical, horizontal, diagonal relationship) matrix problems + 2x1 (A vs. B, C vs. D – horizontal relationship)

2.3 Two strategies:

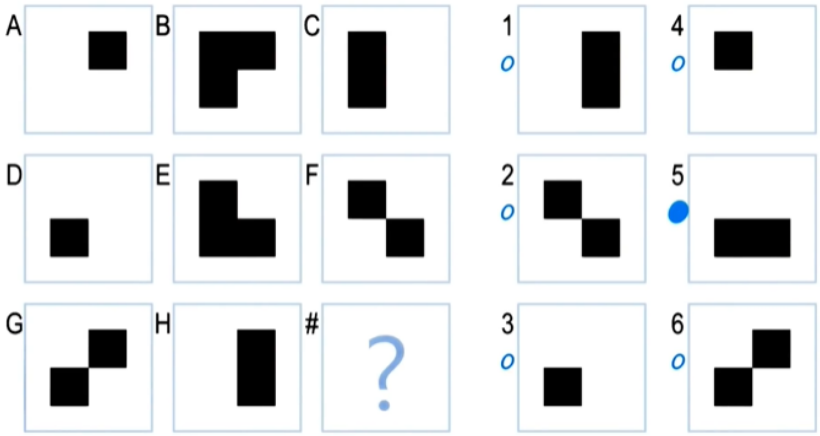
- 2.3.1 Start from the problem and propose a solution – finding a relationship between A and B, and then applying it between C and D
- 2.3.2 Start from the solution – testing each other against the problem and finding out if it matches

2.4 RPM – how to look for differences:

- 2.4.1 Shape – dot, square, triangle etc.
- 2.4.2 Existence – elements appear or disappear
- 2.4.3 Size – elements grow or shrink
- 2.4.4 Rotation – by *n* degrees
- 2.4.5 Reflection – across *x* (horizontal) or *y* (vertical) axis
- 2.4.6 Color – black or white
- 2.4.7 "Completing the overall picture"



2.4.8 OR / XOR relationship



2.5 "(…) in certain level, humans too are just processing signals and inputs in the right way (…) intelligence is hard to define"

2.6 Principles of CS7637:

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

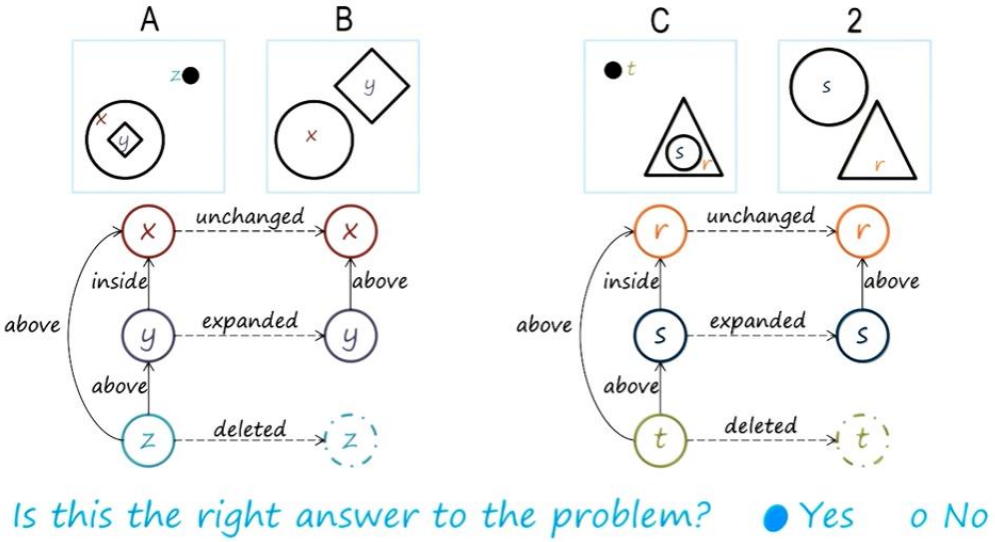
2.7 Frequently-used readings:

- 2.7.1 Winston, P. (1993). Artificial Intelligence (3rd ed.). Addision-Wesley.
- 2.7.2 Stefik, M. (1995). Knowledge Systems. Morgan Kauffman: San Fransisco.
- 2.7.3 Rich, E., & Knight, K. (1991). Artificial intelligence. McGraw-Hill, New York.
- 2.7.4 Russell, S. & Norvig, P. (1995). Artificial Intelligence: A modern approach. Prentice-Hall: Englewood Cliffs.

3. Introduction to CS7637

3.1 Semantic networks – knowledge representation (language which has its vocabulary + content expressed in that language); objects and relationships

- 3.1.1 Lexically – nodes (objects)
- 3.1.2 Structurally – directional links (relationships)
- 3.1.3 Semantically – application-specific labels



3.2 Raven, J. (2003). Raven progressive matrices. In Handbook of nonverbal assessment (pp. 223-237). Springer US.

3.3 Consistent vocabulary

3.4 Winston, P. (1993). Artificial Intelligence (3rd ed.). Addision-Wesley – characteristics of good representations:

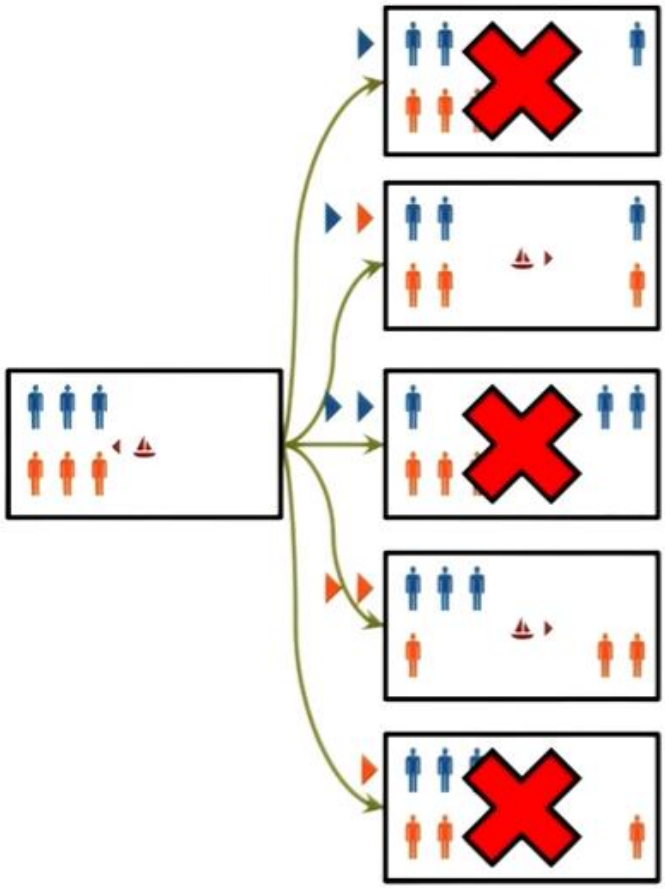
- 3.4.1 Make relationships explicit
- 3.4.2 Expose natural constraints
- 3.4.3 Bring objects and relations together
- 3.4.4 Exclude extraneous details
- 3.4.5 Transparent, concise, complete, fast, computable

3.5 Amarel, S. (1968). On representations of problems of reasoning about actions. Machine intelligence, 3(3), 131-171.

- 3.5.1 Guards & prisoners problem

  Three guards and three prisoners must cross the river, boat may take only one or two people at a time, prisoners may never outnumber guards on either coast, though prisoners may be alone on either coast

  Each node is a state



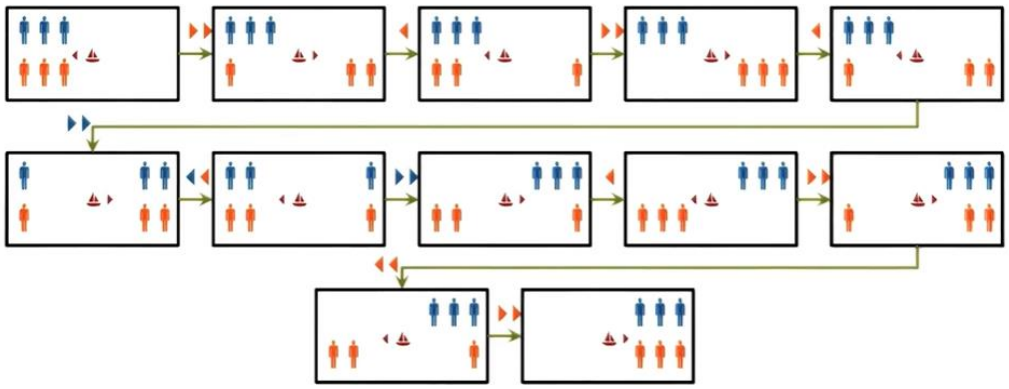
3.5.2 or cannibals and missionaries

3.5.3 or jealous husbands

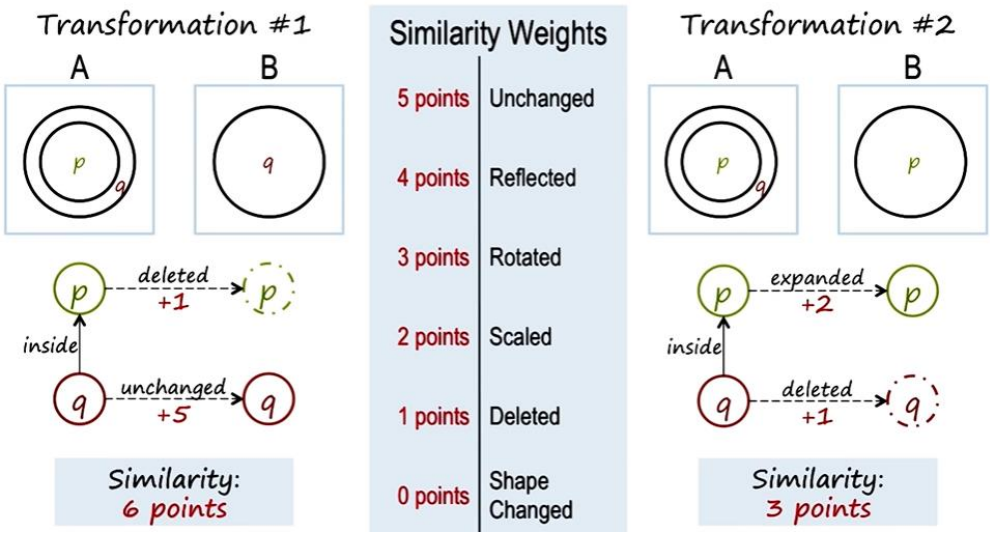
3.5.4 or brothers and sisters

3.6 “If you have the right knowledge representation, problem solving becomes very easy.”

3.7 Full example (guards & prisoners)



3.8 Weights with represent & reason - similarity weights:



3.9 Correspondence problem – given two situations, which object in one situation corresponds to which object in the other situation?

3.10 "In KBAI, and in cognition in general, the focus is always on relationships, not only on objects or the features of these objects."

4. Generate and Test

4.1 Generate and test – problem solving technique – given the problem, generate potential solutions to it and then test their efficiency; not very efficient but can be a very powerful method

4.2 Knowledge-based AI:

- 4.2.1 Knowledge representations
- 4.2.2 Problem solving techniques
- 4.2.3 Architectures

4.3 Dumb generators - dumb testers:

- 4.3.1 Dumb generator – generate a very large number (combinatorial explosion) potential successor states; states might be identical to the initial or previous states
- 4.3.2 Dumb testers dismiss only the illegal states, where prisoners outnumber guards

4.4 Dumb generators - smart testers – identify if the state has already occurred; merges the solutions if they are the same

4.5 Smart generators – dumb testers – generator does not even generate states that have already occurred

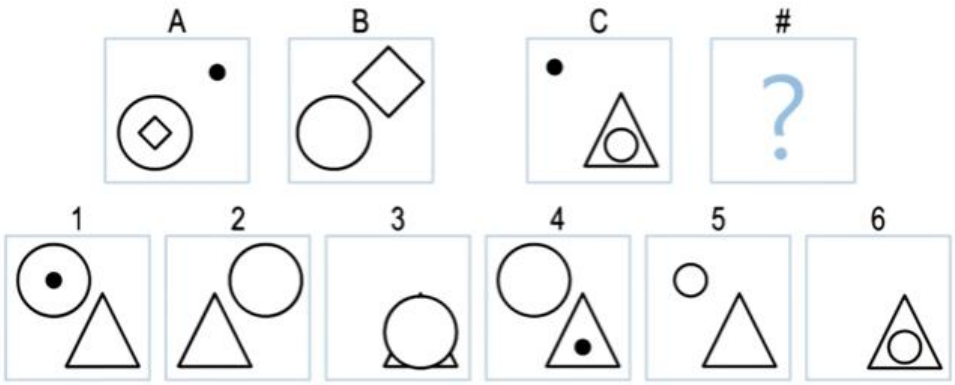
4.6 Smart generators and smart testers – generator: "I don't need to ever generate a state that involves sending one person over to an empty coast"; could be on the tester side too

Smart generators and smart testers are critical when solving RPM problems and this is where semantic networks helps a lot.

4.7 Raven, J. (2003). Raven progressive matrices. In *Handbook of Nonverbal Assessment*. (pp. 223-237). Springer US.

4.8 "Any knowledge representation picks a level of abstraction at which it represents the world."

4.9 Generate and test vs. RPM problems:

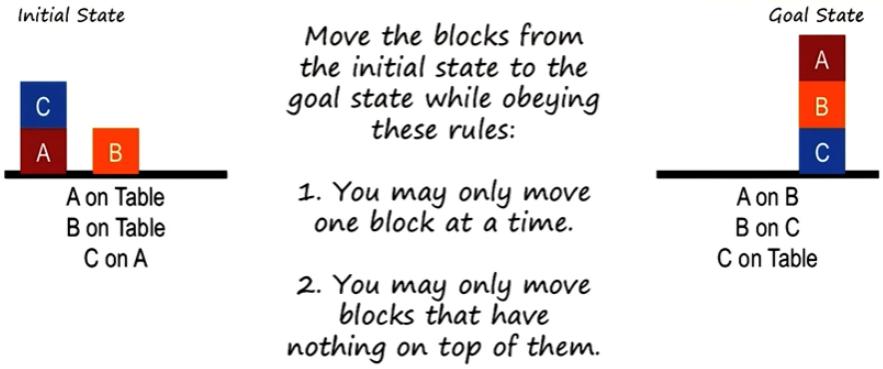


4.9.1 Generate the solution (D) based on the chosen transformation (A, B) and then test it (level of confidence) against each of the given answers (1, 2, 3, 4, 5, 6)

4.9.2 Take each of the given answer (1, 2, 3, 4, 5, 5), describe the transformations (C vs. 1, 2 (…) and test it against the relationship between the given objects (A, B)

5. Means-Ends Analysis

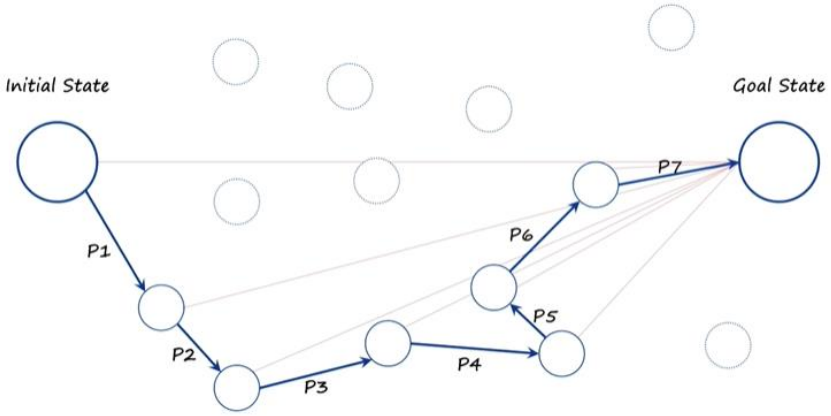
5.1 Winston, P. (1993). Artificial Intelligence (3rd ed.). Addision-Wesley.



5.2 State spaces:

- 5.2.1 Initial state
- 5.2.2 States in between, pathfinding
- 5.2.3 Goal state

5.3 Differences in state spaces:



- 5.3.1 *Ends* – the reduction between the difference with the current state and the goal state
- 5.3.2 *Means* – the application of the operator that will help reduce the difference
- 5.3.3 "At any given state, pick an operator that will help you deduce the difference between the current state and the goal state."

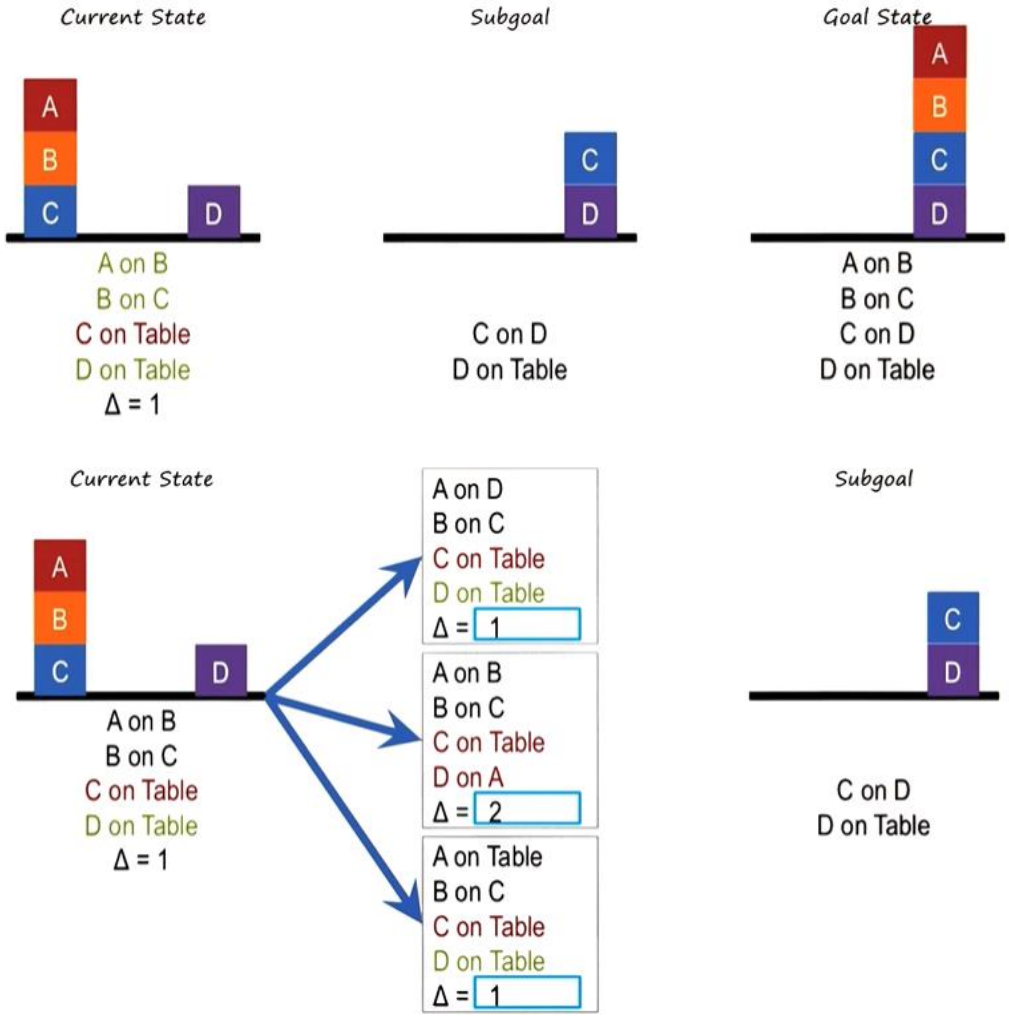
5.4 Means-ends analysis does not always takes us towards the goal. Sometimes it might take us away…or get caught in loops.

5.5 Universal AI methods – generate and test, means-ends analysis:

- 5.5.1 Can't guarantee success
- 5.5.2 Often very costly (computational efficiency)
- 5.5.3 Sometimes called *weak* because they make only little use of knowledge
- 5.5.4 **BUT** can be applied to a very large class of problems

5.6 Problem reduction – decomposing a hard problem into multiple (smaller, simpler) problems; defining sub-goals

s



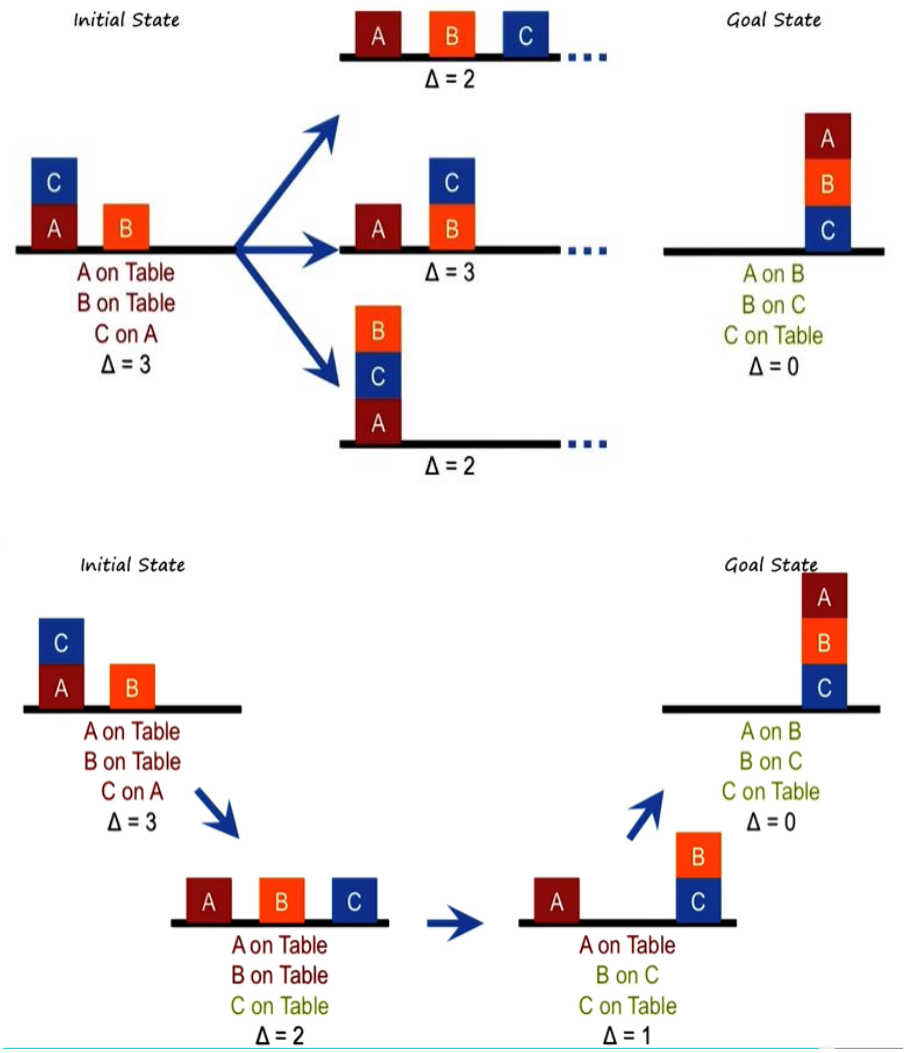
5.7 Means-ends analysis:

5.7.1 For each operator that can be applied:

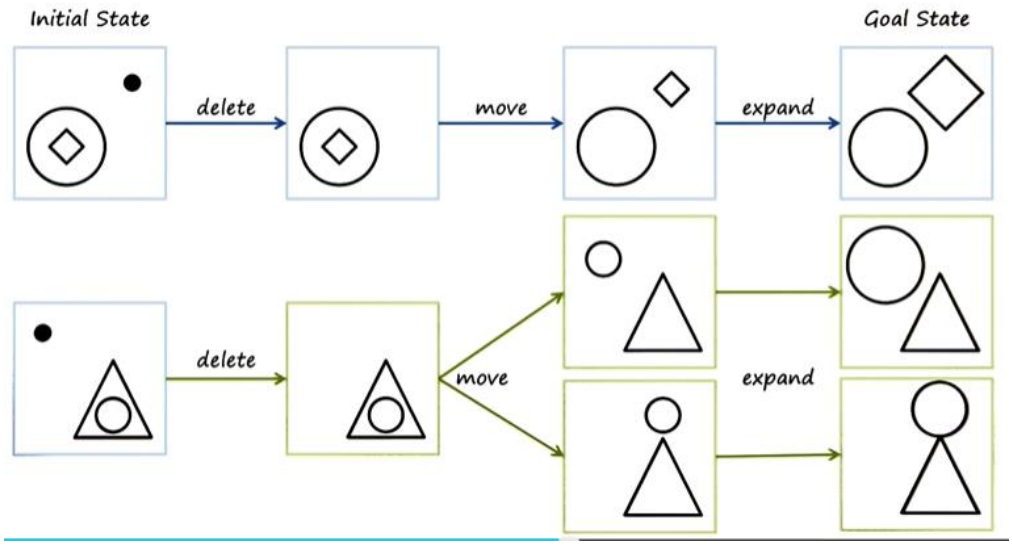
Apply the operator to the current state

Calculate difference between new state and goal state

5.7.2 Prefer state that minimizes distance between new state and goal state



5.8 RPM vs. means-ends analysis, problem reduction and generate and test:



6. Production Systems

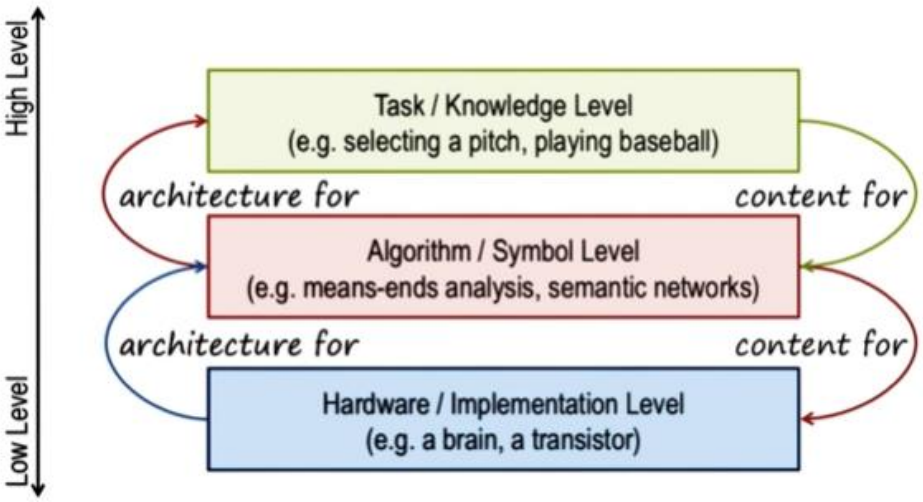
6.1 "Production systems are kind of cognitive architecture in which knowledge is represented in a form of rules."

6.2 Lehman, J. F., Laird, J. E., & Rosenbloom, P. S. (1996). A gentle introduction to Soar, an architecture for human cognition. Invitation to Cognitive Science, 4, 212-249.

6.3 Function of a cognitive architecture:

- 6.3.1 F: P* -> A (Percepts -> Action)
- 6.3.2 Selecting actions – "What should we next?" based on the history of percepts

6.4 Levels of cognitive architectures:

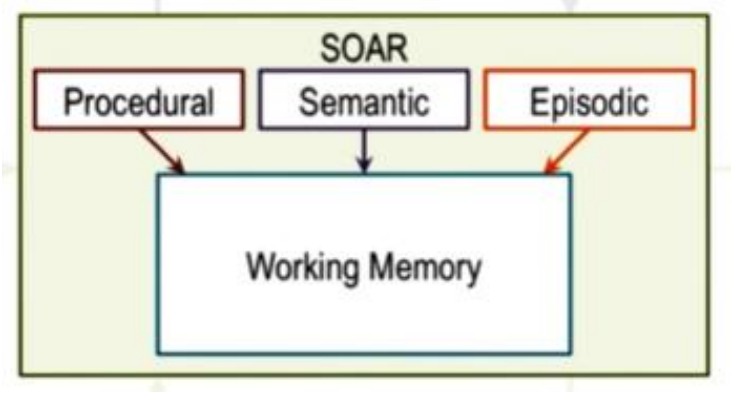


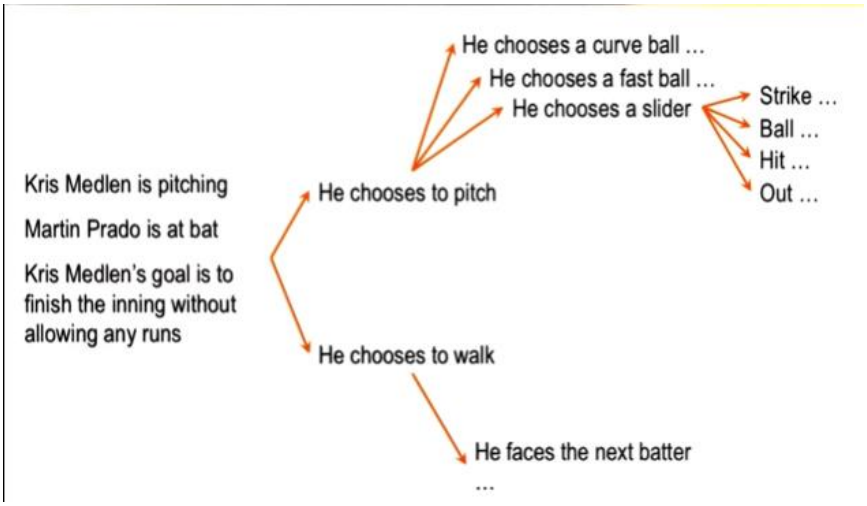
6.5 Assumptions of a cognitive architecture:

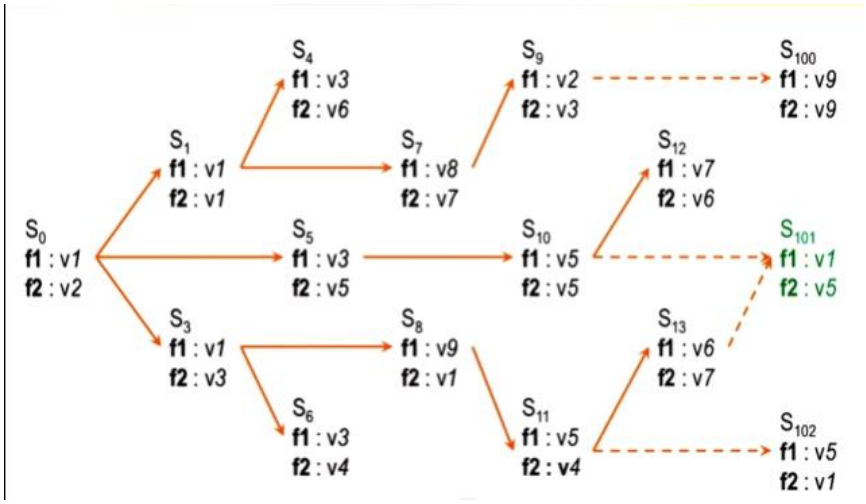
- 6.5.1 Goal-oriented
- 6.5.2 Rich, complex environment
- 6.5.3 Significant knowledge
- 6.5.4 Symbols and abstractions
- 6.5.5 Flexible and function of the environment
- 6.5.6 Constantly learning

6.6 Architecture + Content = Behavior (f: P* -> A)

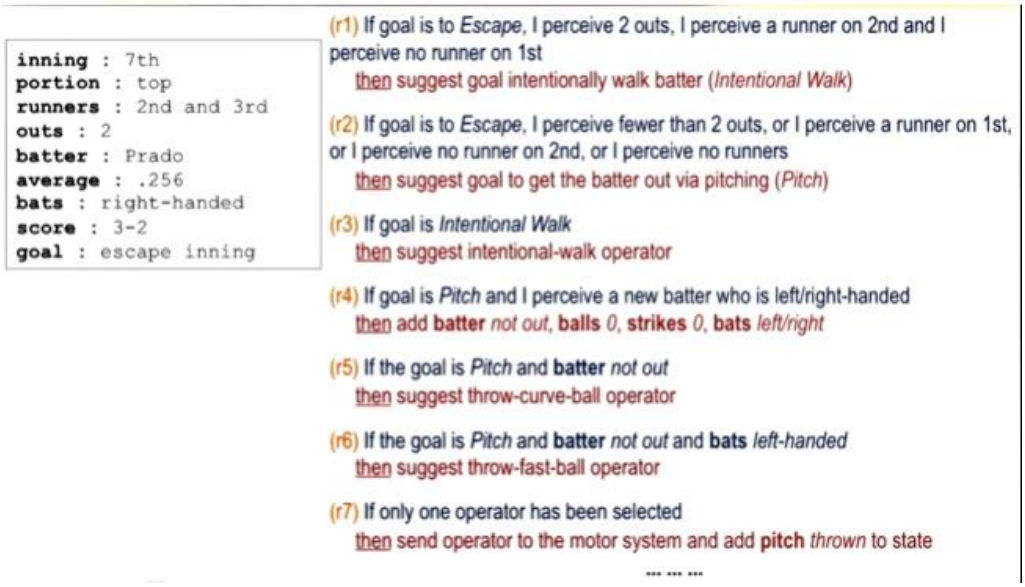
6.7 A cognitive architecture for production systems:

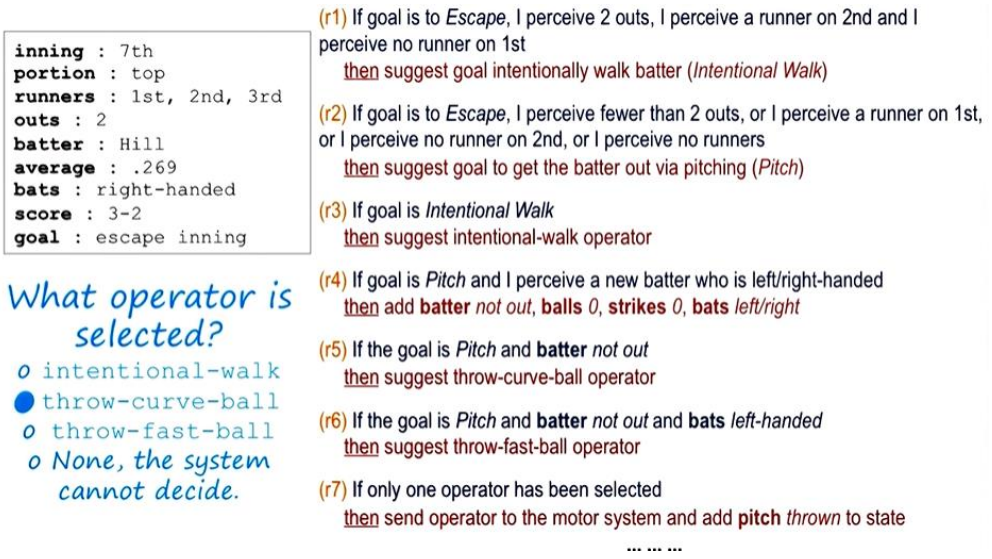






6.8 Production rules that represent procedural knowledge:



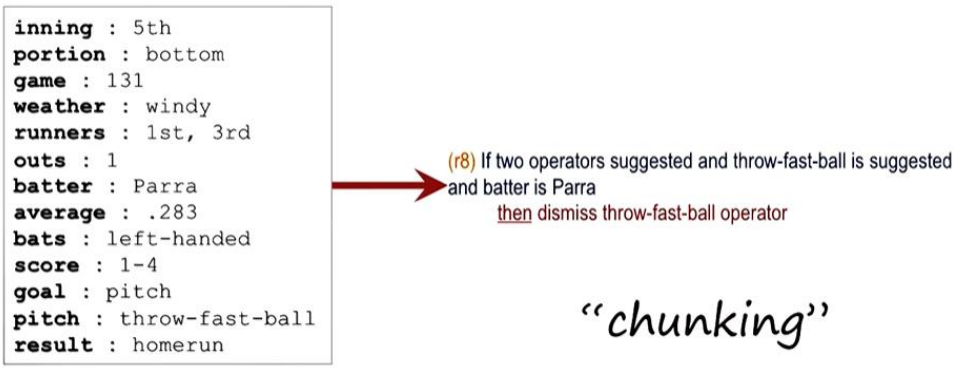


6.9 Chunking:

6.9.1 An impasse occurs when the decision maker cannot make a decision either because not enough knowledge is available or multiple actions are selected and the agent cannot decide

6.9.2 Episodic knowledge about the event

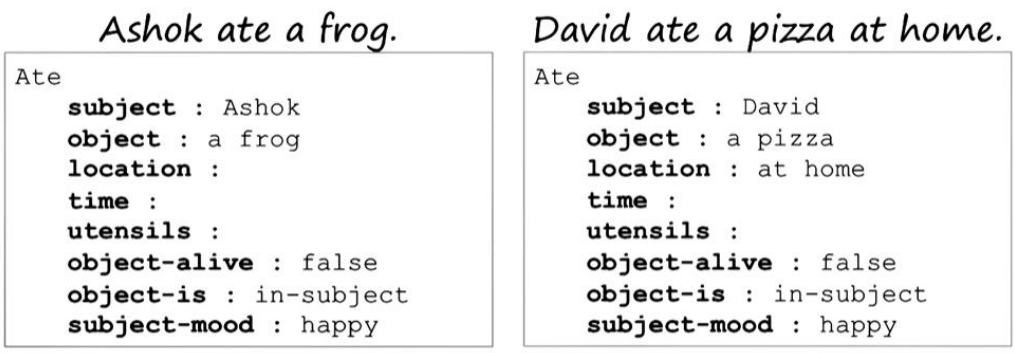
6.9.3 Chunking is a learning technique that SOAR uses to learn rules that can break impasse



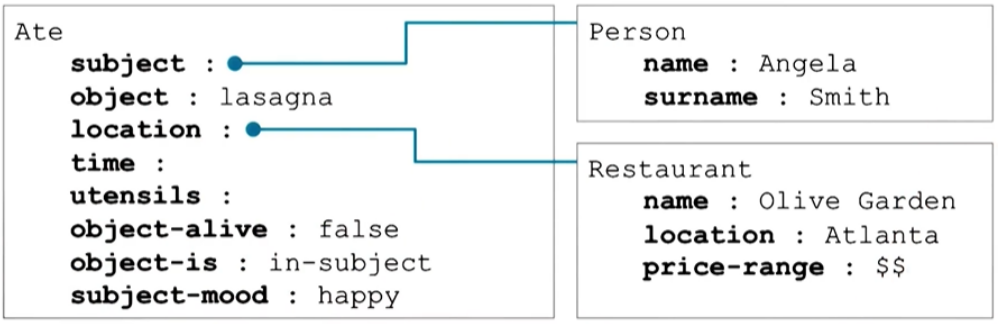
7. Frames

7.1 A frame is a knowledge structure (knowledge representation) that represents stereotypes

7.2 *Ashok ate a frog* exercise – slots and fillers (values);



7.3 Complex frame systems



7.4 Properties of frames:

- 7.4.1 Represent stereotypes
- 7.4.2 Provide default values
- 7.4.3 Exhibit inheritance (classes, sub-classes, instances)

7.5 Frames and semantic networks – RPM case:

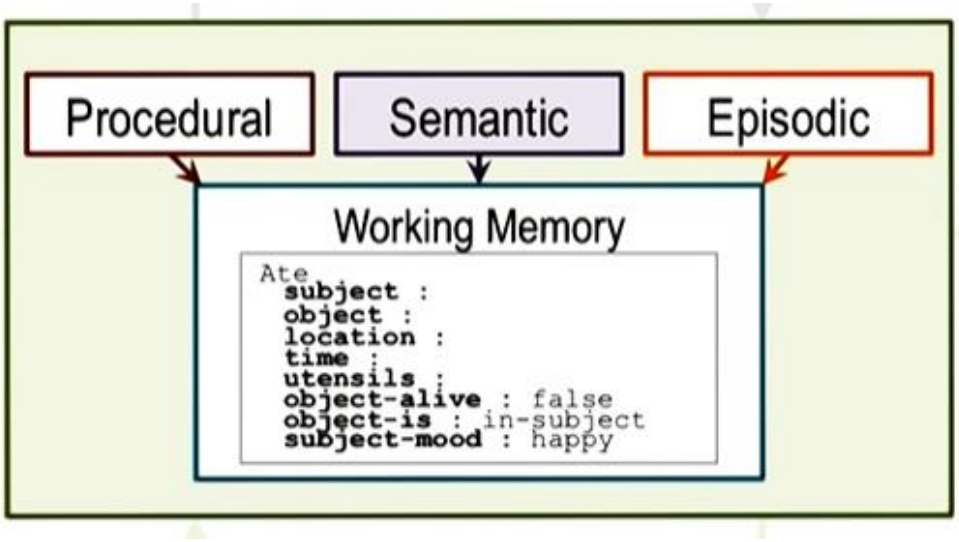
Complete the frame representation of the figure above.

```
Object
    name : x
    shape : triangle
    size : large
    fill : false
    inside :
    above :
```

```
Object
    name : y
    shape : circle
    size : large
    fill : false
    inside :
    above : x
```

```
Object
    name : z
    shape : circle
    size : small
    fill : true
    inside : y
    above : x
```

7.6 Frames and production systems:



7.7 Understanding:

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

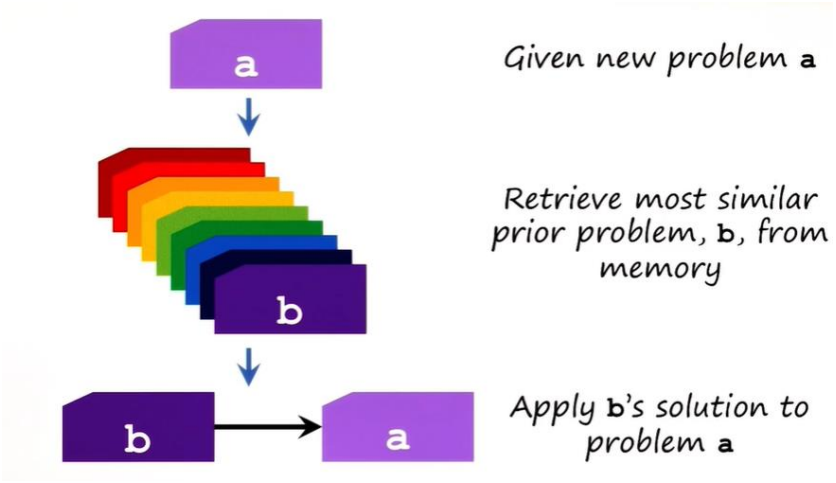
Earthquake

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

8. Learning by Recording Cases

8.1 Learning by recording cases – filing away individual cases we have encountered in the past in order to use them for future problem solving

8.2 Blocks problem:



8.3 Other examples:

- 8.3.1 Tying shoe laces
- 8.3.2 Creating a program in Java
- 8.3.3 Doctor diagnosis

8.4 Case retrieval by nearest neighbor:

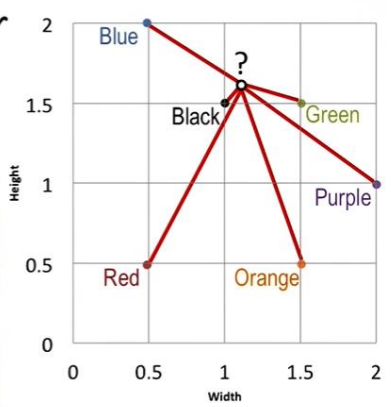
Euclidan distance

Finding the Nearest Neighbor

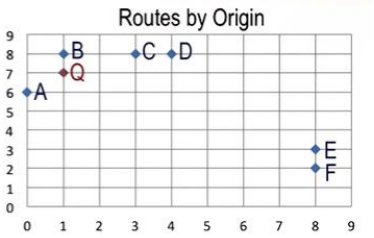
Given existing case at (x_c, y_c) and new problem at (x_n, y_n)

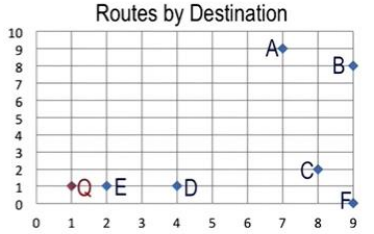
$$d = \sqrt{(y_c - y_n)^2 + (x_c - x_n)^2}$$

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



8.5 Multi-dimensional problem:





D – most similar one to Q problem

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	-

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

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

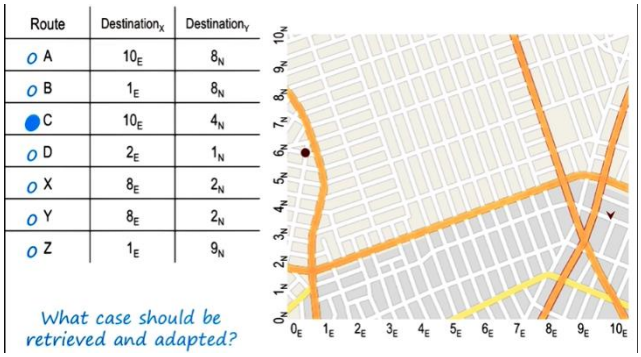
9. Case-Based Reasoning

9.1 “Case-based reasoning builds on learning recording cases. (…) In case-based reasoning, the new problem is similar to previously encountered problem.”

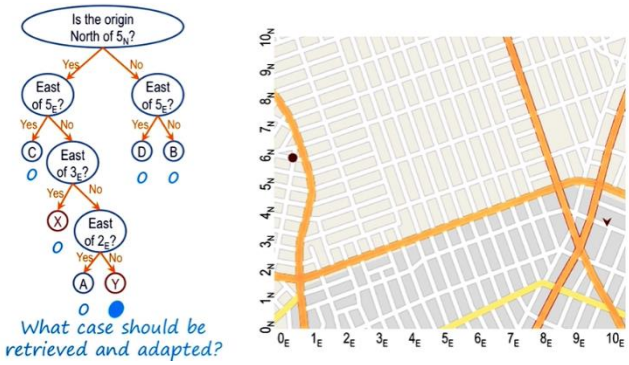
9.2 Phases of case-based reasoning:

9.2.1 Case retrieval – retrieving a case from memory similar to the current problem (kNN)

- A tabular way



- A discrimination tree



9.2.2 Case adaptation – adapting the solution to that case to fit the current problem; example – reading from a file in a program and tweaking this process for the new problem

- Model-based method (example – a map of the world; API of interacting with a particular language)

- Recursive case-based method

Compound analogy – specific type of adaptation by recursive reasoning; composing two partial solutions to get the complete one

- Rule-based method – uses heuristics in the form of rules; heuristics is a rule of thumb that works often but not always (example – looking for a downtown in a new city by searching for the tallest buildings)

Heuristic expressed as a rule example – if you want to make an artifact lighter, try a different material

9.2.3 Case evaluation – evaluating how well the adapted solution addresses the current problem

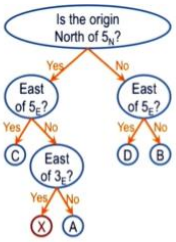
- Simulation

9.2.4 Case storage – storing the new problem and solution as a case

- By index – an index is like a tag that "becomes a particular way of identifying each individual case"; we want an indexical structure which allows for effective and efficient retrieval; O(n) linear

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	$\mathbf{4_E}$	$\mathbf{6_N}$

- By discrimination tree – a knowledge structure in which the cases themselves are the leaf nodes of the tree– "by asking a question we are quickly able to prune away one part of the tree, that makes a search process much more efficient"; O(n) logarithmic



9.3 Assumptions (not always valid) of case-based reasoning:

9.3.1 Patterns exist in the world

9.3.2 Similar problems have similar solutions – example of a situation in which it is not valid can be using touch screens – some of them react to one finger, some can react to using two or three fingers

9.4 "(…) all designers redesign. Design is fundamentally evolutionary. We take all designs and we evolve them slightly and that's how we get a new design."

9.5 Advanced case-based reasoning:

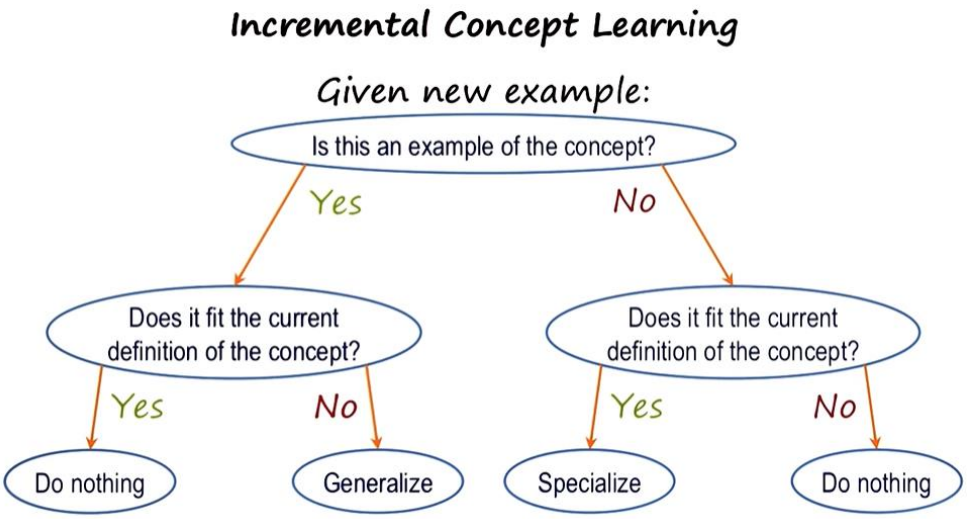
9.5.1 "Evaluation found the solution failed; try adapting again."

9.5.2 "Evaluation found the solution failed; try retrieving a different solution."

9.5.3 "The retrieved solution could not be adapted; retrieve a different solution."

10. Incremental Concept Learning

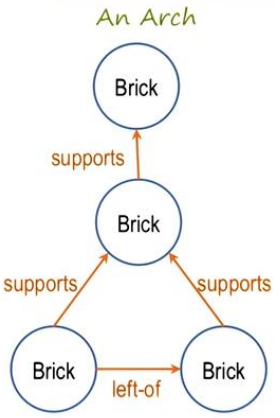
10.1 Example:



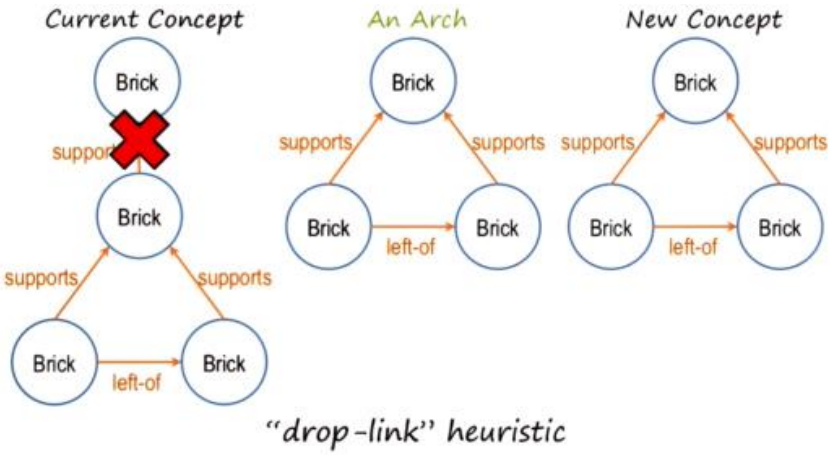
10.2 Some lessons:

- "Learning is often incremental. We learn from one example at a time."
- "Often the examples that we get are labeled."
- "The examples can come in a particular order. (..) The first example typically is a positive example."
- "Abstracting concepts from the examples."
- "The number of examples from which we're extracting concepts is very small."
- "When we're trying to abstract concepts from examples, then what exactly to abstract, what exactly to learn, what exactly to generalize becomes a very hard problem" – tendency to overgeneralize (when seeing a positive example) or to overspecialize (when seeing a negative one)

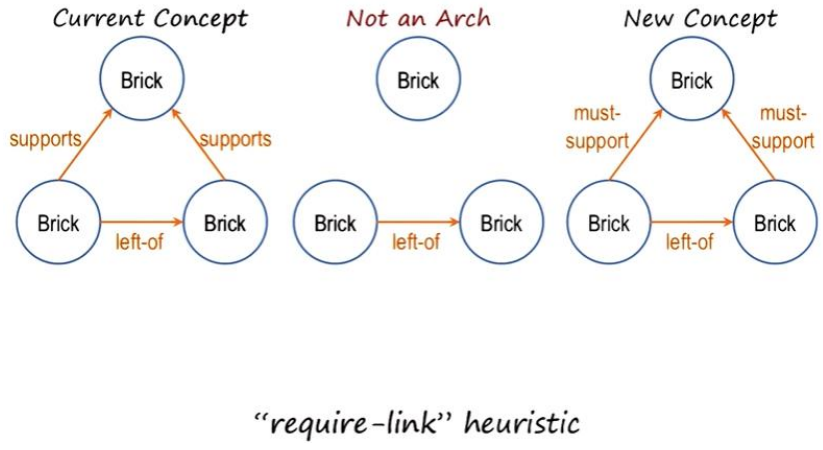
10.3 Variabilization:



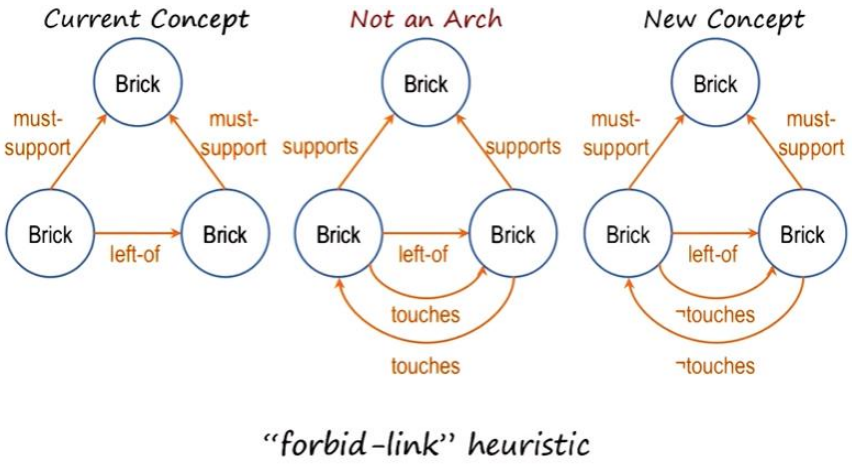
10.4 Generalization to ignore features – "drop-link" heuristic:



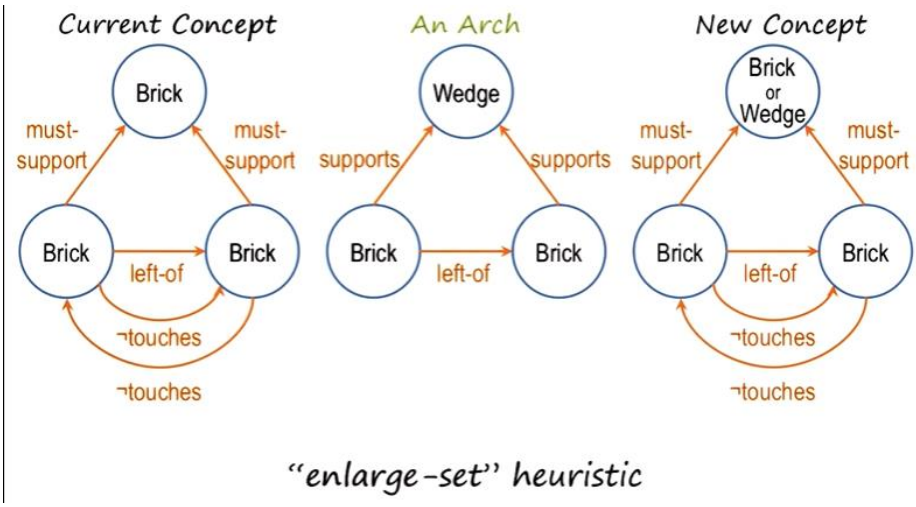
10.5 Specialization to require features – "require-link" heuristic:



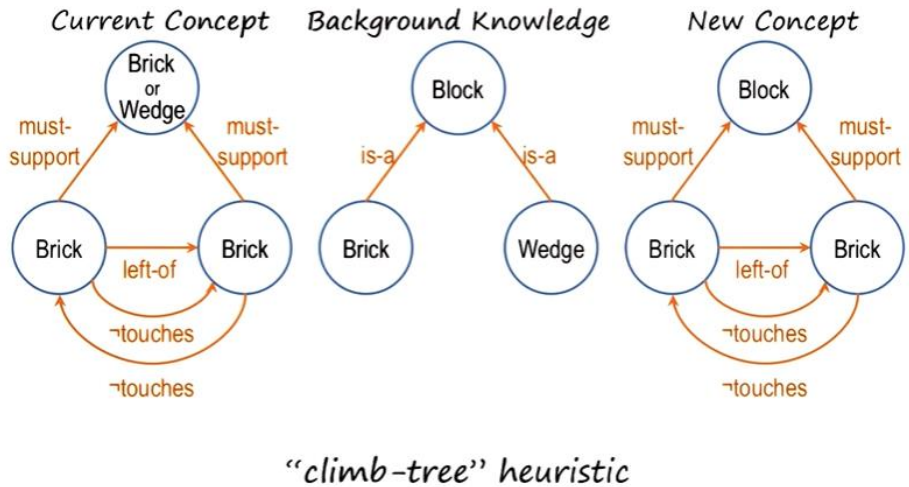
10.6 Specialization to exclude features – "forbid-link" heuristic:



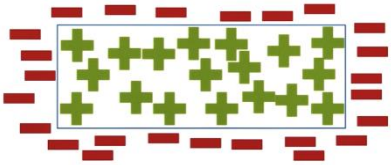
10.7 Generalization to abstract features – "enlarge-set" heuristic:



10.8 Generalization with background knowledge:



10.9 "Given a small set of positive and negative examples, the number of dimensions in which the algorithm can do generalization and specialization is very large." – a concept definition"



10.10 Heuristics for specializing and generalizing:

Heuristics for Specializing and Generalizing

require-link: link must be present to be a positive example of the concept

enlarge-set: multiple objects or links may fit one role in the concept

forbid-link: link must be absent to be a positive example of the concept

climb-tree: generalize over multiple objects in the same role based on knowledge

drop-link: link is not necessary to be a positive example of the concept

close-interval: expand range of values to be a positive example of the concept

10.11 "What we learn in the end depends upon what type of knowledge we begin with."

10.12 "If the example is a negative instance, and the current definition of the concept already excludes it, we don't have to do anything."

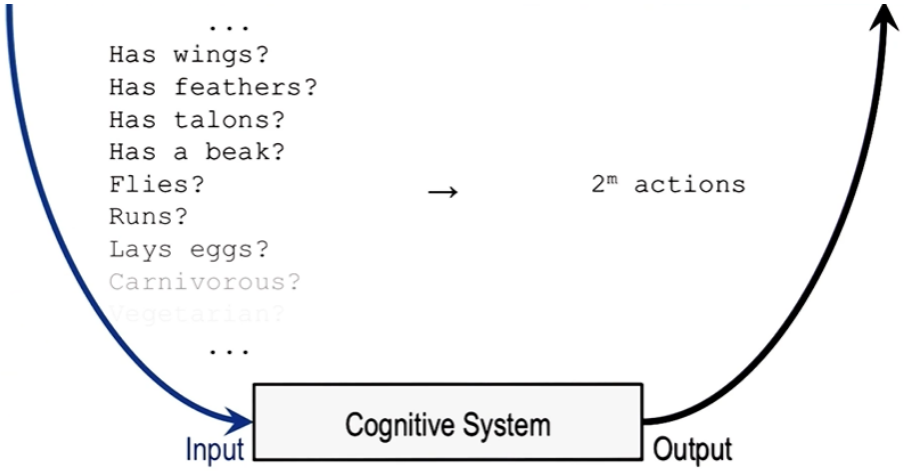
11. Classification

11.1 Stefik, Chapter 7, Part 1 Pgs 543-556

11.2 Stefik, Chapter 7, Part 2 T-Square Resources Pgs 588-596

11.3 Classification – "mapping sets of percepts in the world into equivalence classes, so that we can take actions in the world in an efficient manner."

11.4 Classification in a cognitive system – 2^n percepts -> 2^m actions:



10 percepts = 1024 combinations
100 percepts = 1.2×10^{30} combinations
= 1.2 nonillion combinations
300 percepts = 2.0×10^{90} combinations
= more combinations than atoms in the universe

11.5 Equivalence classes – 2^n percepts -> k concepts -> 2^m actions

11.6 "That's the power of knowledge. When you have it, you can take some complex problem, and break it into a large number of smaller, simpler problems."

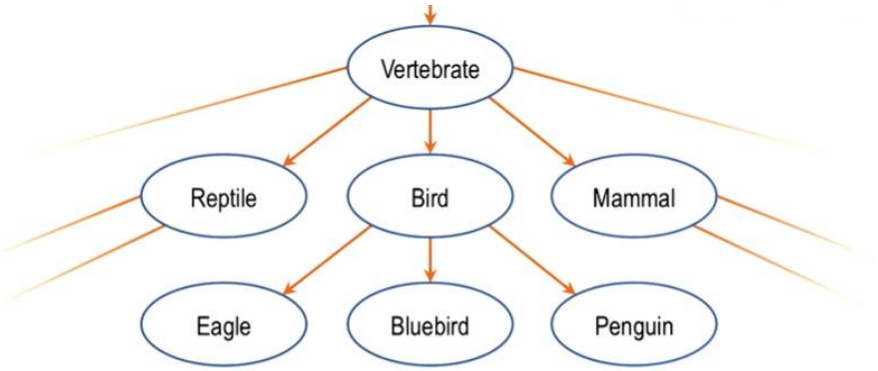
11.7 Example:

For each of these three animals, choose the value for each percept that applies to that animal.

Eagle	Bluebird	Penguin
Lays eggs? ● Yes ○ No ○ Maybe	Lays eggs? ● Yes ○ No ○ Maybe	Lays eggs? ● Yes ○ No ○ Maybe
Has wings? ● Yes ○ No ○ Maybe	Has wings? ● Yes ○ No ○ Maybe	Has wings? ● Yes ○ No ○ Maybe
Has talons? ● Yes ○ No ○ Maybe	Has talons? ● Yes ○ No ○ Maybe	Has talons? ○ Yes ● No ○ Maybe
Flies? ● Yes ○ No ○ Maybe	Flies? ● Yes ○ No ○ Maybe	Flies? ○ Yes ● No ○ Maybe
Has fur? ○ Yes ● No ○ Maybe	Has fur? ○ Yes ● No ○ Maybe	Has fur? ○ Yes ● No ○ Maybe
Large? ● Yes ○ No ○ Maybe	Large? ○ Yes ● No ○ Maybe	Large? ● Yes ○ No ○ Maybe

11.8 Concept hierarchies – establish and refine (top-down) approach

- "organization is power"
- well-suited for one kind of organization of concepts
- well-suited for one set of situations where we know something already we are to break and what we're trying to establish



11.9 "The idea that we can decide on the features that should go into a super class, given the features that are shared among the subclasses does not work for all the concepts. It particularly works for concepts that have a formal nature."

11.10 Types of concepts:

- Axiomatic (more formal) – set of necessary and sufficient conditions that we all agree with; well defined and easier to program; example: a circle
- Prototype – base concept defined by some typical properties with overridable properties; might be represented by a frame; example: a chair
- Exemplar (less formal) – concepts defined by implicit abstractions of instances, or exemplars, of the concept; can be culture- or individual-specific; example: beauty

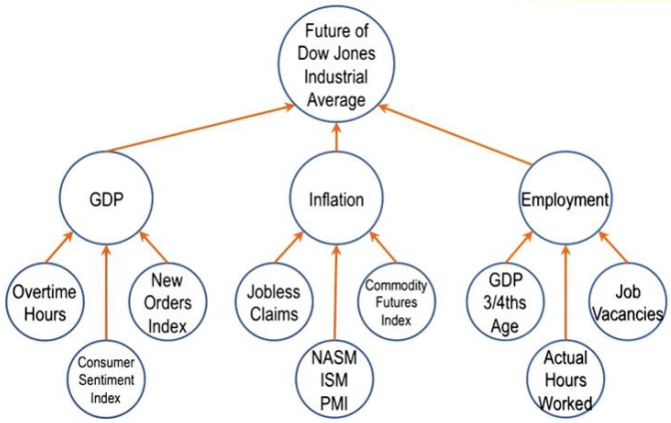
- Qualia – raw sensations that we may get from our sensors; example: bitterness

11.11 Order of concepts:

Rank the following concepts based on their formality:

1. Inspirational
2. Reptile
3. Foo
4. Right Triangle
5. Holiday
6. Saltiness

11.12 Bottom-up search – the task is to make a prediction at a root node, based on the leaf nodes – identify and abstract approach:



12. Logic

12.1 Logic – a formal language that allows us to make assumptions about the world in a very precise way; forms the basis of planning

12.2 AI agent and formal logic would consist of two parts:

- 12.2.1 Knowledge base – agent's knowledge about the world in the form of sentences in the language of logic
- 12.2.2 Inference engine – applies rules of inference to the knowledge of the agent

12.3 Properties of inferencing:

- 12.3.1 Soundness – only valid conclusions can be proven
- 12.3.2 Completeness – all valid conclusions can be proven

12.4 Predicate – a function that maps object arguments to true or false values; examples:

- *Feathers(bluebird)*
- "If an animal has feathers, then it is a bird" – implicative relationship
  If Feathers(animal) :
  Then Bird(animal)

12.5 Conjunctions and disjunctions:

- "If an animal lays eggs and it flies, then it is a bird."
  Lays-eggs(animal) ^ *Flies (animal) :*
  Then Bird(animal)
- "If an animal lays eggs or it flies, then it is a bird."
  Lays-eggs(animal) v *Flies (animal) :*
  Then Bird(animal)
- "If an animal flies and is not a bird, it is a bat."
  Flies(animal) ^ ¬ *Bird(animal) :*
  Then Bat(animal)

12.6 "Implies that" symbol (=>)
Lays-eggs(animal) ^ *Flies(animal)* => *Bird(animal)*

12.7 Notation equivalency:

Operator	Symbol	Accepted Symbol
AND	A ∧ B	A & B A && B
OR	A ∨ B	A | B A || B
NOT	¬A	!A ~A
IMPLIES	A ⇒ B	A = B A == B A => B

12.8 Truth tables:

A	B	A ∨ B
True	True	True
True	False	True
False	True	True
False	False	False

A	B	C	A ∨ (B ∧ ¬C)
True	True	True	True
True	True	False	True
True	False	True	True
True	False	False	True
False	True	True	False
False	True	False	True
False	False	True	False
False	False	False	False

12.9 Commutative property – A ^ B = B ^ A

12.10 Distributive property – mixture of operators – A ^ (B v C) = (A ^ B) v (A ^ C)

12.11 Associative property – same operators – A v (B v C) = (A v B) v C

12.12 de Morgan's Law – ¬(A ^ B) = ¬A v ¬B

12.13 Truth of implications – "whether or not the implication holds true:

Truth of Implications

A	B	A ⇒ B
True	True	True
True	False	False
False	True	True
False	False	True

12.14 Implication elimination – A => B = ¬A v B

12.15 Rules of inference – instantiate general rules to prove specific claims

12.15.1 Modus Ponens

Sentence 1: p => q

Sentence 2: p

inferring Sentence 3: q

12.15.2 Modus Tollens:

Sentence 1: p => q
Sentence 2: ¬q
inferring Sentence 3: ¬p

12.16 Proving:

12.16.1 Truth tables
12.16.2 Modus Ponens or Modus Tollens

12.17 Propositional logic – also called the zeroth order logic, no variables

12.18 Universal quantifiers – predicate calculus (first-order logic):

12.18.1 Universal quantifier:

For one animal:

$$\text{Lays-eggs(animal)} \land \text{Flies(animal)} \Rightarrow \text{Bird(animal)}$$

For all animals:

$$\forall x[\text{Lays-eggs}(x) \land \text{Flies}(x) \Rightarrow \text{Bird}(x)]$$

"Universal Quantifier"

12.18.2 Existential quantifier:

For at least one animal:

$$\exists y[\text{Lays-eggs}(y) \land \text{Flies}(y) \Rightarrow \text{Bird}(y)]$$

"Existential Quantifier"

12.19 Resolution Theorem Proving – computationally efficient method

12.19.1 Convert every sentence into a conjunctive normal form – it can have literals, disjunction literals or a conjunction of disjunctional request – bootstrapping of the robot's knowledge base (axiom that the robot assumes to be true)

12.19.2 Reflect on the percepts

12.19.3 Negate what the robot wants to prove

We know:
~~S1: ¬can-move ⇒ ¬liftable~~

By implication elimination:
S1: can-move ∨ ¬liftable

We find:
S2: ¬can-move

We assume:
S3: liftable

How do we prove the box is not liftable?

~~S1: can-move ∨ ¬liftable~~
~~S2: ¬can-move~~
~~S3: liftable~~

12.20 More complex example:

We know:
~~S1: ¬can-move ∧ battery-full ⇒ ¬liftable~~

By implication elimination:
~~S1: ¬(¬can-move ∧ battery-full) ∨ ¬liftable~~

By deMorgan's Law:
S1: can-move ∨ ¬battery-full ∨ ¬liftable

We find:
S2: ¬can-move
S3: battery-full

How do we prove the box is not liftable?

A null condition (contradiction):

~~S1: can-move ∨ ¬liftable ∨ ¬battery-full~~
~~S2: ¬can-move~~
~~S3: battery-full~~
~~S4: liftable~~

12.21 Horn clause – a disjunction that contains at most one positive literal (example – S1 above)

12.22 The cognitive connection:

12.22.1 Deduction – reasoning from causes to effects

12.22.2 Abduction – reasoning from effects to causes

12.22.3 Induction – given some relationship between cause and effect (for example) how do we generalize it between a cause and effect relationship for a population

13. Planning

13.1 Block world problem – how can we find which goal to select among the various goals that are available

13.2 States – painting a ceiling problem:

Initial State:
On(Robot, Floor) ∧
Dry(Ladder) ∧
Dry(Ceiling)

Goal State:
Painted(Ceiling) ∧
Painted(Ladder)

13.3 Operators can be applied if and only if the pre-conditions of the operator are true in the world; examples: climb-ladder, paint-ceiling, descend-ladder, paint-ladder:

climb-ladder:
Precondition:
On(Robot, Floor) ∧
Dry(Ladder)
Postcondition:
On(Robot, Ladder)

paint-ceiling:
Precondition:
On(Robot, Ladder)
Postcondition:
Painted(Ceiling) ∧
¬Dry(Ceiling)

descend-ladder:
Precondition:
On(Robot, Ladder) ∧
Dry(Ladder)
Postcondition:
On(Robot, Floor)

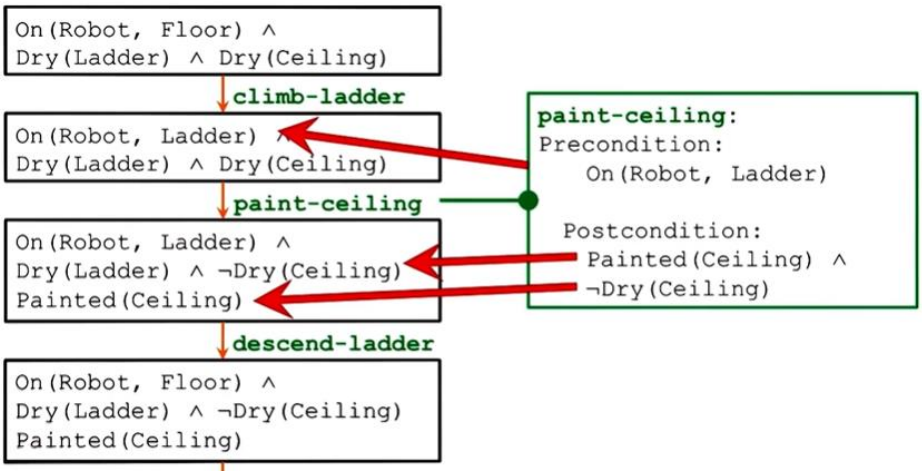
paint-ladder:
Precondition:
On(Robot, Floor)
Postcondition:
Painted(Ladder) ∧
¬Dry(Ladder)

13.4 Pre-conditions can only have positive literals, as opposed to post-conditions

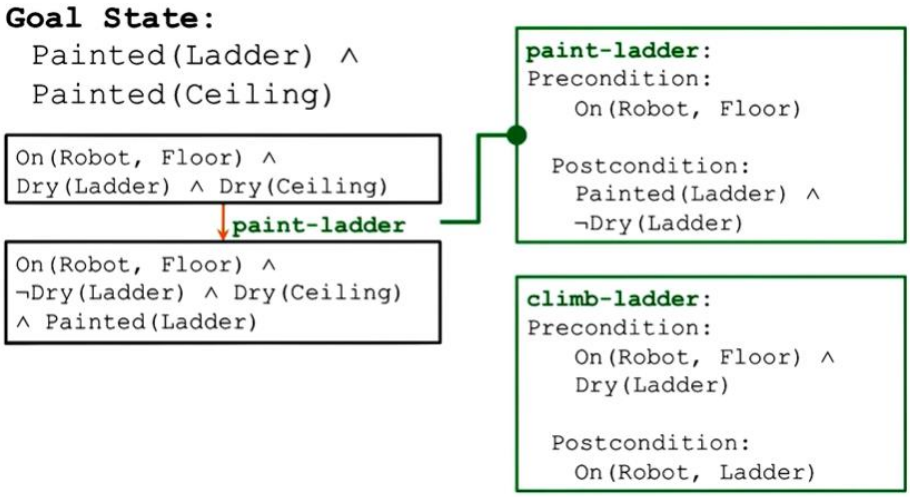
13.5 Planning and state spaces:

- 13.5.1 Means-ends analysis – goals provide us with the control knowledge of deciding how to select between different operators
- 13.5.2 Intelligent agents map perceptual history into actions
- 13.5.3 Planning uses more systematic method by dealing with action selection or with operator selection
- 13.5.4 Operators – mental representation of actions available in the world

13.6 An example of a plan:

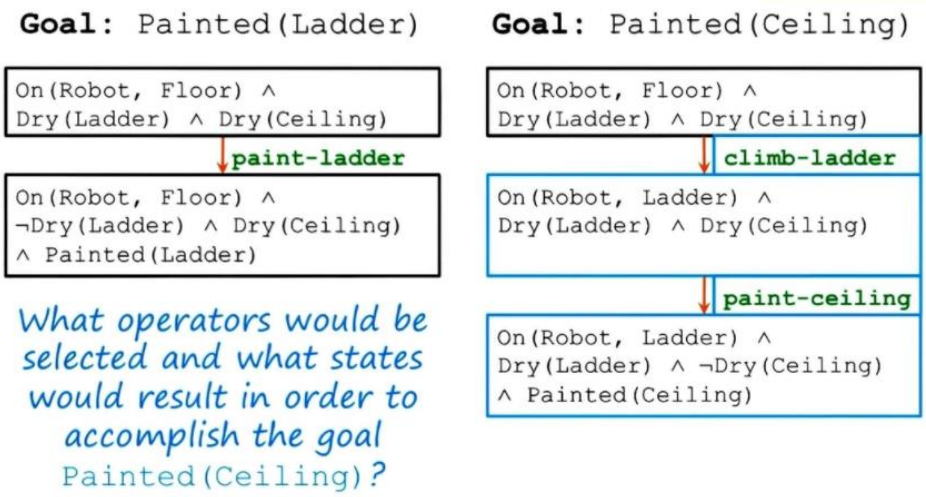


13.7 Means-ends analysis conflict – linear planner (doesn't try to reason about the conflict between the goals):



13.8 Partial order (non-linear) planning occurs with the multiple goals, when the plan for achieving one goal clobbers another goal. Example: couch assembled before putting it in the apartment, then – a need to disassemble it because it does not fit into the doors and assembling it again, after it is in the apartment.

13.9 Problem reduction – identifying sub-goals to achieve the goal:



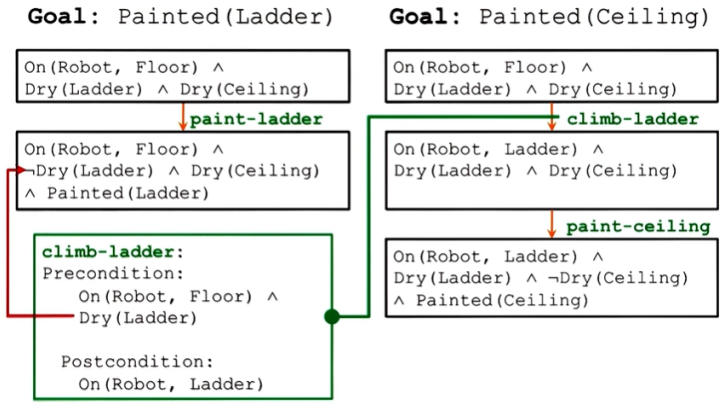
13.10 Detecting conflicts:

13.10.1 Viewing two goals as if they were independent of each other and coming up with a partial plan for each of these goals

13.10.2 Detection:

```
For each precondition in current plan:
   If precondition for an operator in the
   current plan is clobbered by a state in
   another plan:
      Promote current plan's goal above other
      plan's goal
```

13.10.3 Example – promote *Painted(Ceiling)* above *Painted(Ladder)*:

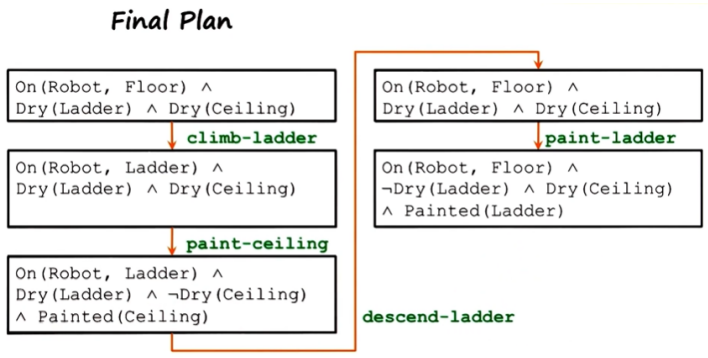


13.11 Open pre-conditions:

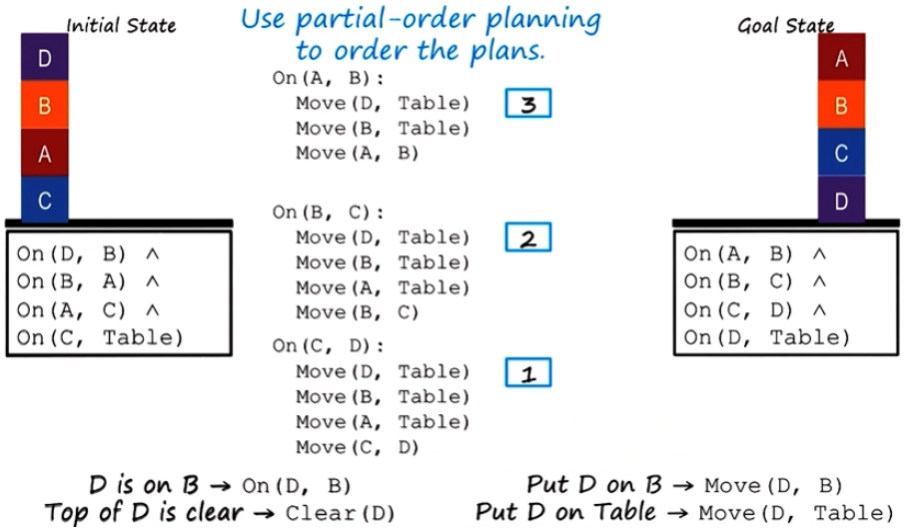
13.11.1 "Knowledge is not only about the world. Knowledge is also control knowledge that helps us select between operators."

13.11.2 "Goals provide control knowledge."

13.11.3 "We can view partial order planning as an interaction between several different kinds of agents or abilities. Each agent represents a small, micro-ability."

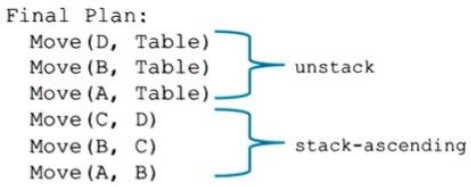


13.12 Block world example:



13.13 Hierarchical task network (HTN) planning:

13.13.1 Decomposition – macro-operators:



13.13.2 Examples – unstack and stack-ascending:

unstack:

```
Precondition:
  On(w, x) ∧
  On(x, y) ∧
  On(y, z) ∧
  On(z, Table)
Postcondition:
  On(w, Table) ∧
  On(x, Table) ∧
  On(y, Table) ∧
  On(z, Table)
Method:
  Move(w, Table)
  Move(x, Table)
  Move(y, Table)
```

stack-ascending:

```
Precondition:
  On(a, Table) ∧
  On(b, Table) ∧
  On(c, Table) ∧
  On(d, Table)
Postcondition:
  On(a, b) ∧
  On(b, c) ∧
  On(c, d) ∧
  On(d, Table)
Method:
  Move(c, d)
  Move(b, c)
  Move(a, b)
```

13.13.3 “Intelligent agents addresses complex problems with thinking at multiple levels of abstraction so that at any level of abstractions the problem appears small and simple.”

14. Understanding

14.1 Understanding relies heavily on frames.

14.2 Stories provide structure of data:

Story 1:
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.

Story 2:
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.

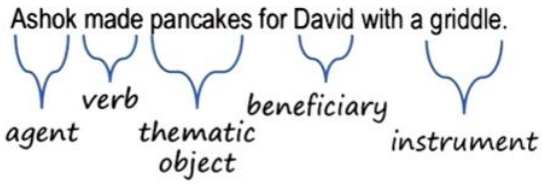
Focus on sentences:

Sentence 1:
A serious earthquake killed 25 people in Lower Slabovia.

Sentence 2:
The President of Lower Slabovia killed 25 proposals for earthquake prediction.

14.3 Thematic role pertains to the relationship of various words in the sentence to this particular action of making.

Example – "Ashok made pancakes for David with a griddle.":




```
Thematic Role
  verb : make
  agent : Ashok
  beneficiary : David
  thematic object : pancakes
  instrument : griddle
```

14.4 Thematic role system – a type of a frame system where the frame represents an action or event identified by a verb; generating expectations

14.5 Constraints:

David went to the meeting with Ashok by car.

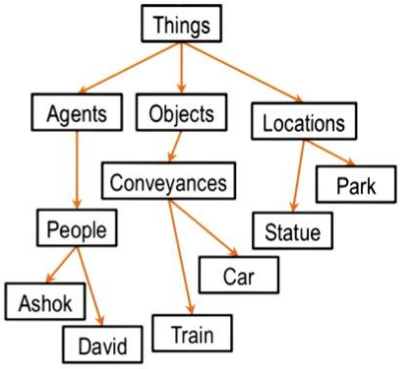
```
Thematic Role
  verb : go
  agent : David
  coagent : Ashok
  destination : meeting
  conveyance : car
```

Prepositional Constraints

Preposition	Thematic Roles
by	agent, conveyance, location
for	beneficiary, duration
from	source
to	destination
with	coagent, instrument

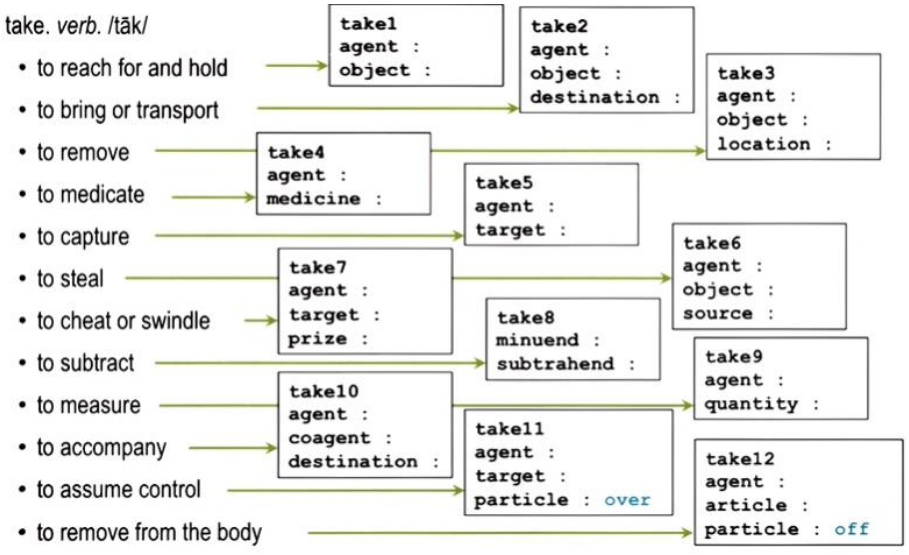
14.6 Resolving ambiguity in case of prepositions, top-down processing:

Ontology – "a particular theory about the nature of being or the kinds of things that have existence"; in our case – the categories of the words

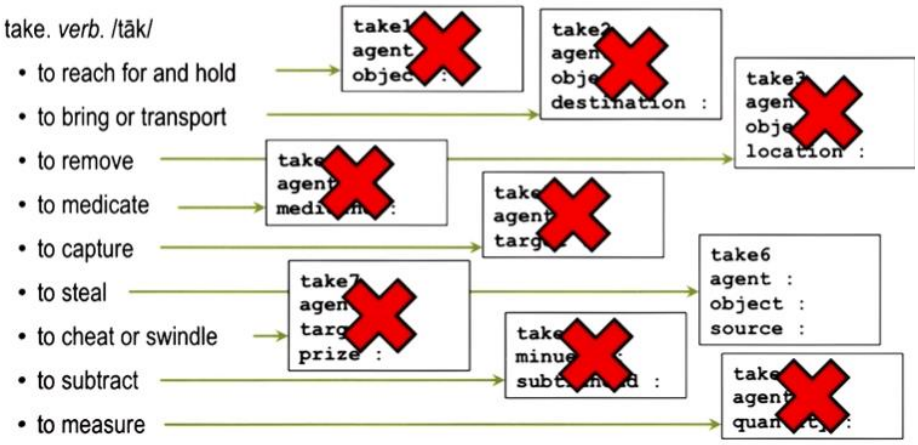


14.7 Resolving ambiguity in case of verbs:

Take:



I took the candy from the baby.



14.8 The earthquake sentences:

Sentence 1: A serious earthquake killed 25 people in Lower Slabovia.

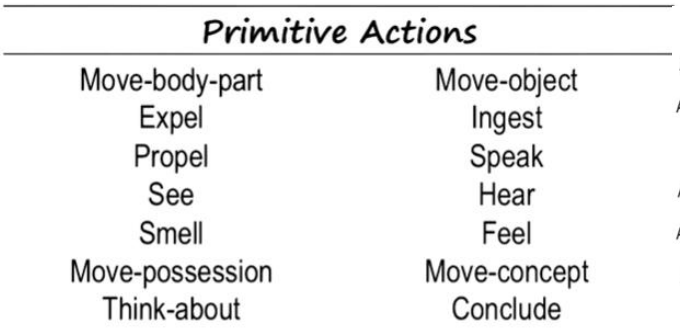
Sentence 2: The President of Lower Slabovia killed 25 proposals for earthquake prediction.

15. Common sense reasoning

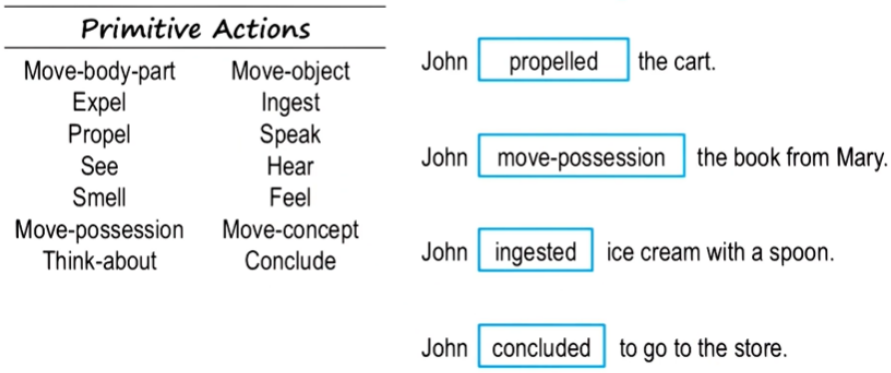
15.1 Different words – same meaning:

Ashok **ate** a frog.	Ashok **partook** of a frog.
Ashok **devoured** a frog.	Ashok **swallowed** a frog.
Ashok **consumed** a frog.	Ashok **dined** on a frog.
Ashok **ingested** a frog.	Ashok **inhaled** a frog.

15.2 Primitive actions:



pushed – took – ate – decided



15.3 Thematic roles and primitive actions – structured knowledge representation:

John pushed the cart.

```
Action Frame
  primitive : propel
  agent : John
  object : cart
```

15.4 Implied actions:

~~John fertilized the field.~~

John put fertilizer on the field.

```
Action Frame
  primitive : move-object
  agent : John
  object : fertilizer
  destination : field
```

Bill shot Bob.

Bill propelled a bullet into Bob.

```
Action Frame
  primitive : propel
  agent : Bill
  object : bullet
  destination : Bob
```

15.5 Actions and sub-actions:

Ashok put the wedge on the block.

```
Action Frame
  primitive : move-object
  agent : Ashok
  object : wedge
  destination : block
  sub-action :
  sub-action :
  sub-action :
```

```
Action Frame
  primitive : move-body-part
  object : fingers
  destination : closed
```

```
Action Frame
  primitive : move-body-part
  object : hand
```

```
Action Frame
  primitive : move-body-part
  object : fingers
  destination : open
```

15.6 State changes:

Ashok enjoyed eating the frog.

```
Action Frame
  primitive : ingest
  agent : Ashok
  object : frog
  result :
```

```
State-Change Frame
  object : Ashok's mood
  destination : happy
```

Complete the frame representations for these sentences.

Anika decided to have a glass of water.

```
● Action o State Change
  primitive : conclude
  agent : Anika
  result :
```

```
● Action o State Change
  primitive : ingest
  agent : Anika
  object : glass of water
```

Marc loved watching TED talks.

```
● Action o State Change
  primitive : see
  agent : Marc
  object : TED talks
  result :
```

```
o Action ● State Change
  object : Mark's mood
  destination : happy
```

15.7 Actions and resultant actions:

Maria told Ben to throw the ball.

```
Action Frame
  primitive : speak
  agent : Maria
  result :
```

```
Action Frame
  primitive : propel
  agent : Ben
  object : ball
```

15.8 Theory of mind

16. Scripts

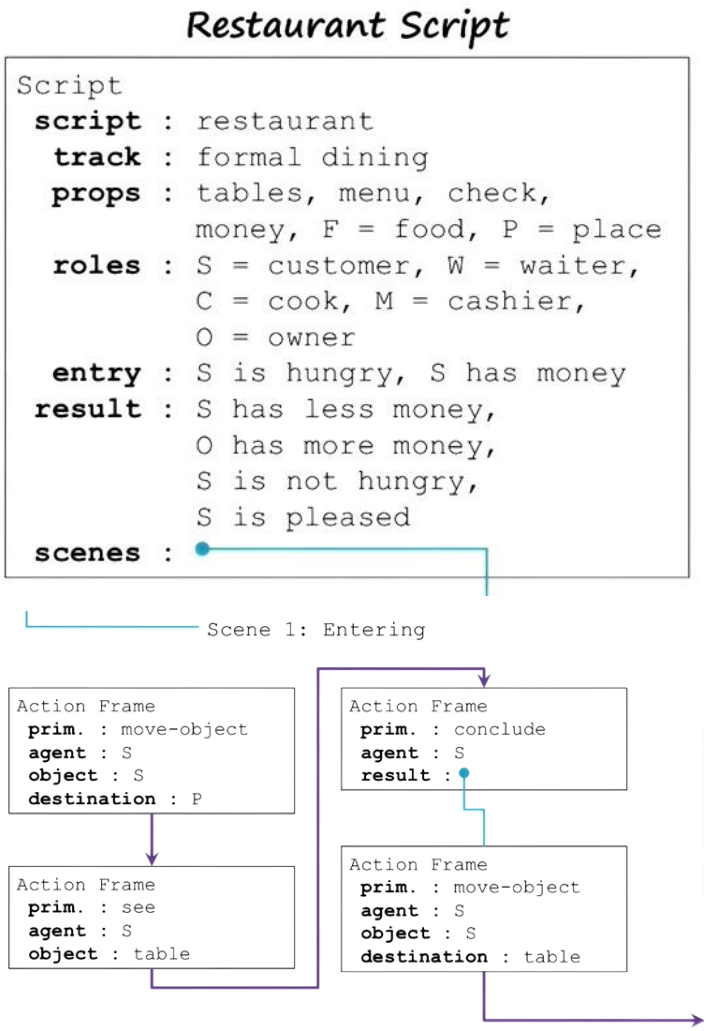
16.1 "Stories help you make sense of the world. They help you generate expectations, even before events occur."

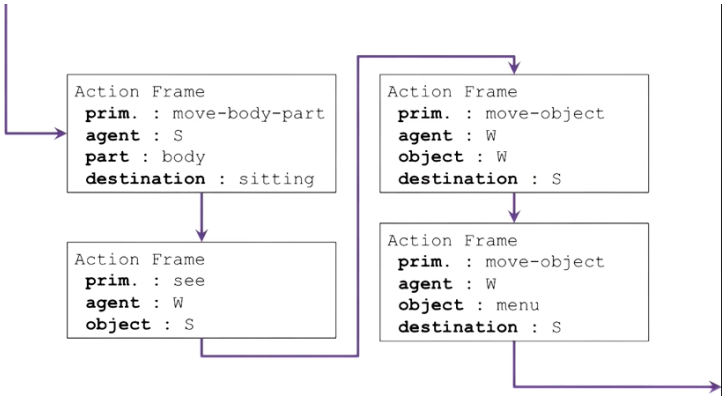
16.2 Scripts help to generate expectations about scenes in the world – coffee house example

16.3 Script - a knowledge representation for capturing causally (each event sets off, or causes, the next event) coherent (the causal connections between events make sense) set of events (the parts are actions or scenes in the world)

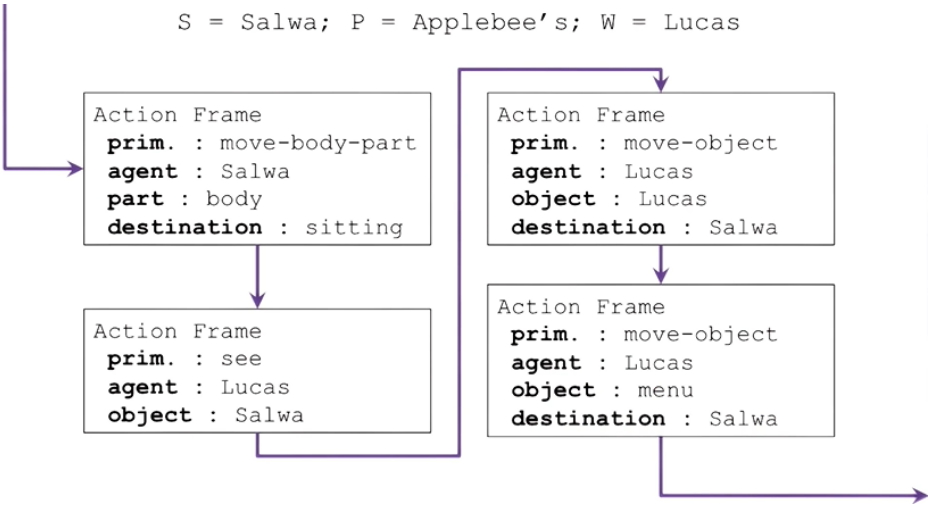
16.4 Parts of a scripts:

- 16.4.1 Entry conditions – conditions necessary to execute the script
- 16.4.2 Result – conditions that will be true after the script has taken place
- 16.4.3 Props – objects involved in the execution of the script
- 16.4.4 Roles – agents involved in the execution of the scripts
- 16.4.5 Track – variations or "subclasses" of the particular script
- 16.4.6 Scenes – the sequence of events that occurs during execution of the script



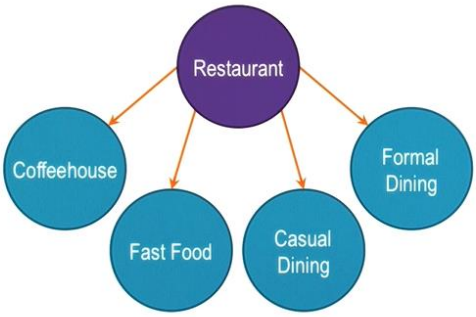


16.5 Form vs. content:



16.6 "A surprise, it has two sides to it. One, when things do not happen the way we expect them to happen. Two, when things happened the way in which we were not expecting them to happen."

16.7 Tracks:



16.8 Learning a script:

Which of the following prior topics might help an agent learn a script? Check all that apply.

- [x] Semantic Networks
- [x] Frames
- [ ] Production Systems
- [ ] Learning by Recording Cases
- [x] Incremental Concept Learning
- [x] Planning
- [x] Commonsense Reasoning

16.9 Using a script:

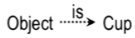
Which of the following prior topics might help an agent use a script? Check all that apply.

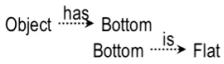
- [ ] Generate & Test
- [ ] Means-Ends Analysis
- [x] Problem Reduction
- [ ] Case-Based Reasoning
- [x] Classification
- [x] Logic
- [x] Understanding

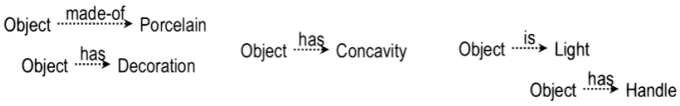
17. Explanation-based learning

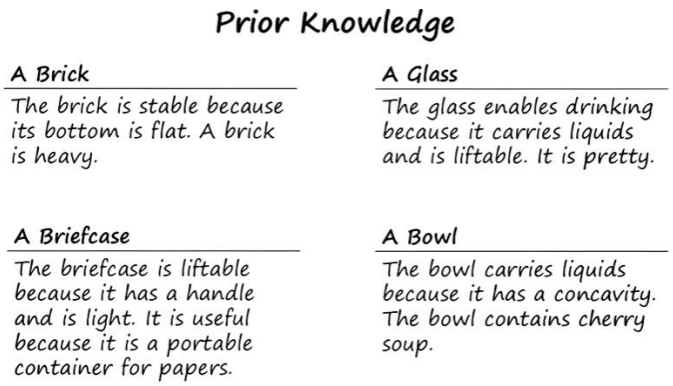
17.1 Explanation-based learning – an agent does not learn about new concepts but about connections among existing parts

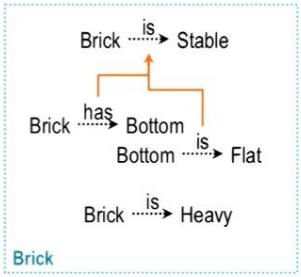
17.2 Concept space and prior knowledge:

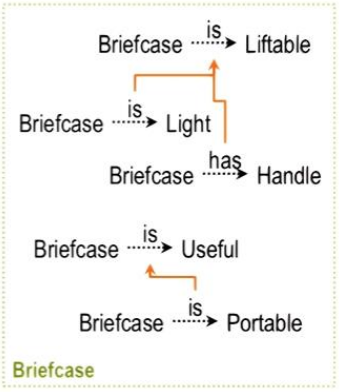


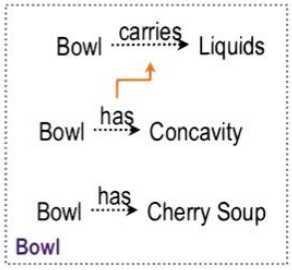


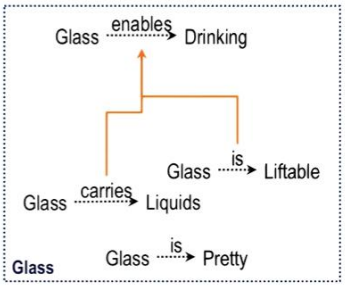




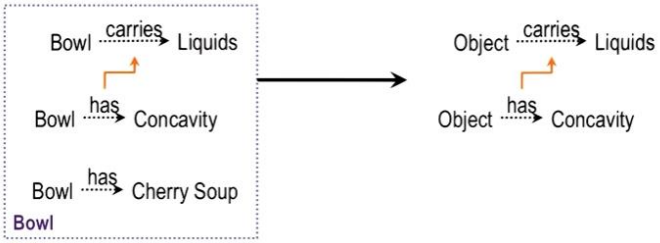




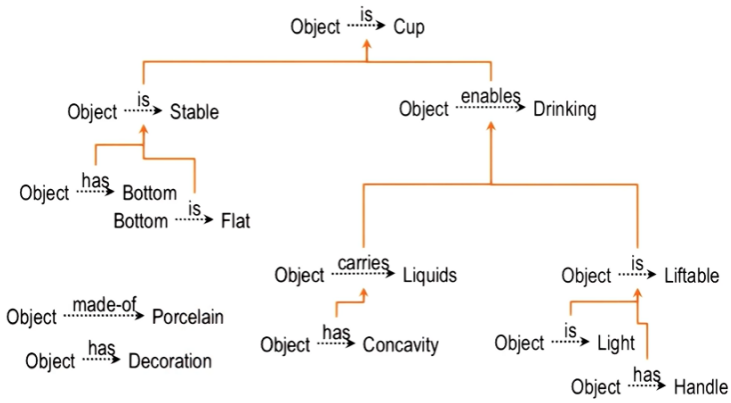




17.3 Abstraction – only things that are in fact causally related



17.4 Transfer – connects with problem reduction and the notion of planning:



17.5 "It's not a question of putting a lot of knowledge into a program. Instead, the real question is, in order to accomplish a goal, what is the minimal amount of knowledge that the AI agent actually needs?"

17.6 Speed-up learning – using existing concepts in new ways, finding connections by building explanations (novel situations -> creative solutions)

18. Analogical reasoning

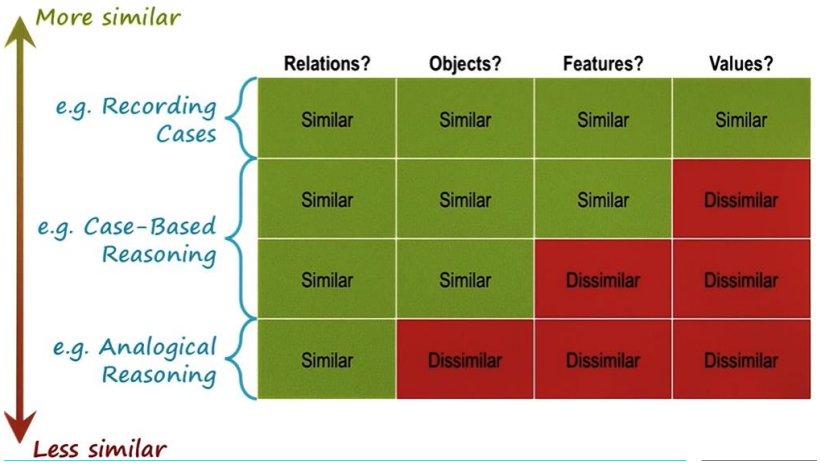
18.1 Similarity:

18.1.1 Between objects, relationships, features and values of features

18.1.2 Measures of similarity notion in k-nearest neighbors, case-based reasoning (array, discrimination tree)

18.2 Cross-domain analogy – physician who treats a patient's tumor with a laser vs. the strategy of the army that uses decomposition as a strategy to capture the castle

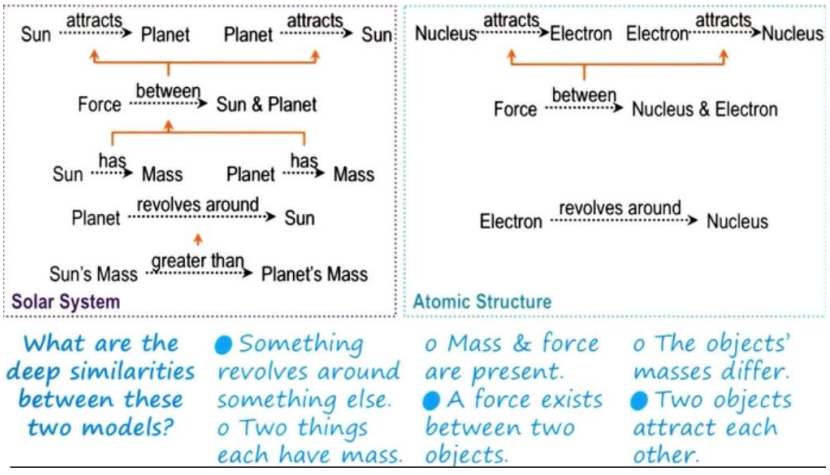
18.3 Spectrum of similarity:



18.4 Five stages of analogical reasoning:

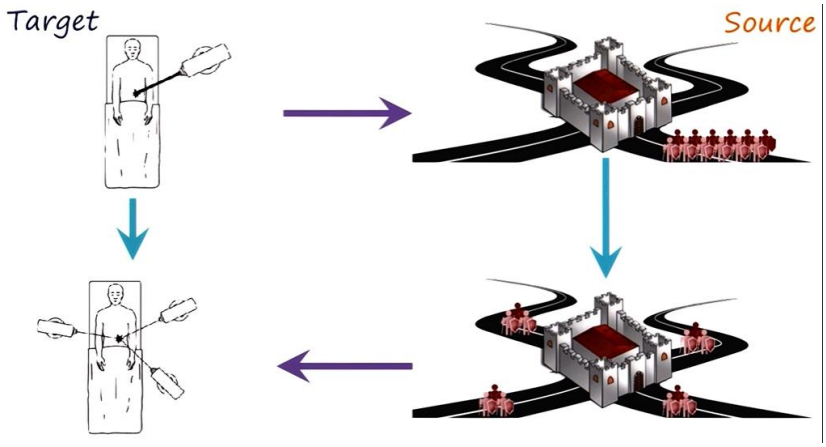
18.4.1 Retrieval

- Superficial similarity – features, counts, objects
- Deep similarity – relationships between objects, relationships between relationships



18.4.2 Mapping – what in the target problem corresponds to the source case (making use of higher order relationships; example: describing a laser and an army as resources)

18.4.3 Transfer – depends on the correct mapping



18.4.4 Evaluation – example: simulation

18.4.5 Storage – evaluated case might become a source case for a new target problem that comes later; incremental learning

18.5 Types of similarity:

18.5.1 Semantic – conceptual similarity between the target problem and the source case

18.5.2 Pragmatic – similarity of external factors, such as goals

18.5.3 Structural – similarity between representational structures

18.6 Design by analogy – biomimicry as an example:

18.7 Design by analogy retrieval – structure-behavior-function model

18.7.1 Function represented by initial and goal state achieved by a behavior

18.7.2 Behavior represented as a series of states and transitions between those states

18.8 Sometimes there is a loop between mapping and transfer or if evaluation fails – between evaluation and transfer/mapping or retrieval

18.9 Advanced questions in analogical reasoning:

- 18.9.1 Common vocabulary
- 18.9.2 Abstraction and transformation
- 18.9.3 Compound and compositional analogies
- 18.9.4 Visuospatial analogies
- 18.9.5 Conceptual combination

19. Version spaces

19.1 Version spaces is closely related to incremental concept learning

19.1.1 Similarity – small set of examples, arriving one at a time

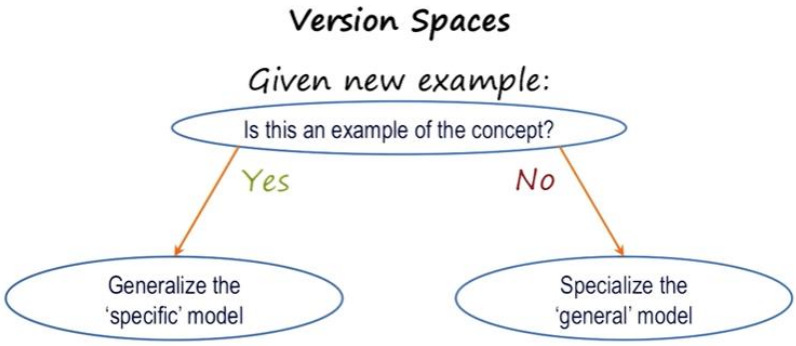
19.1.2 Differences:

- background knowledge may not be available
- we may have no control over the ordering of the examples
- for version spaces, positive examples lead to generalization, negative ones to specialization

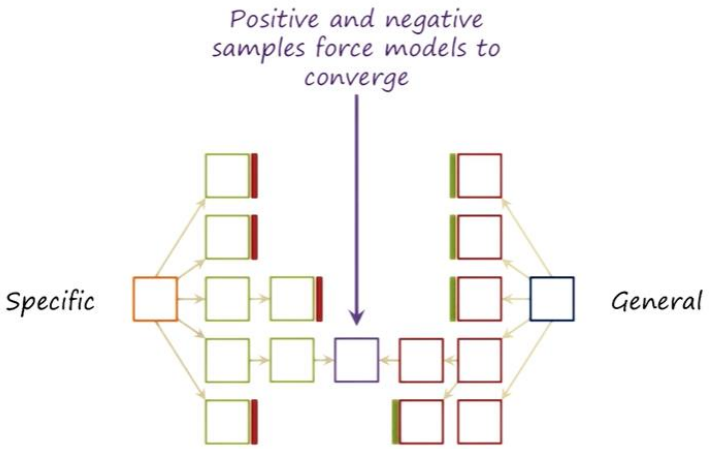
19.2 Abstract version spaces:

19.2.1 Most specific model matches exactly one thing

19.2.2 Most general model matches everything



Prune models that no longer match the data. Then:



19.3 Model – a representation of the world such that there is 1:1 correspondence between what is being represented to the world and the representation itself

19.4 Algorithm for version spaces:

Algorithm for Version Spaces

For each example:

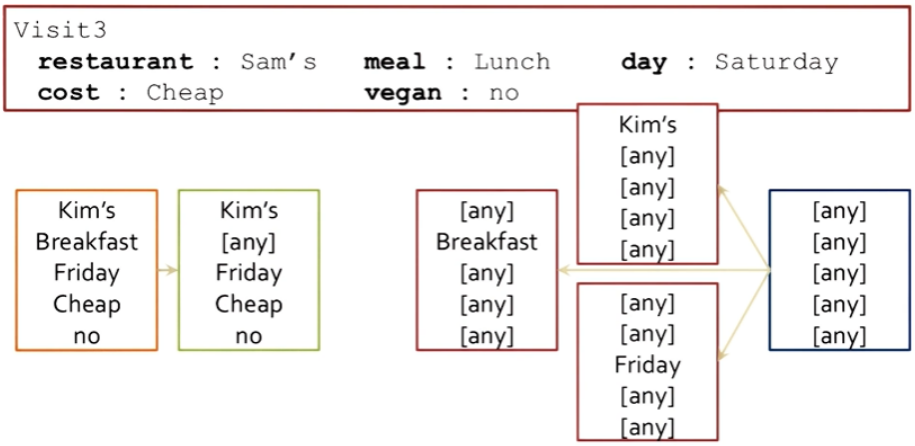
If the example is **positive**:
Generalize all **specific** models to include it
Prune away **general** models that cannot include it

If the example is **negative**:
Specialize all **general** models to include it
Prune away **specific** models that cannot include it

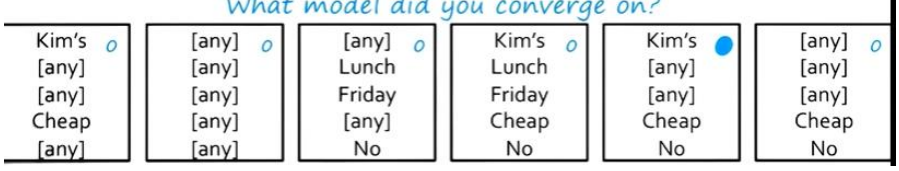
Prune away any models **subsumed** by other models

Importance of convergence

19.5 Example:



Number	Meal	Meal	Day	Cost	Vegan	Reaction?
Visit1	Kim's	Breakfast	Friday	Cheap	No	Yes
Visit2	Kim's	Lunch	Friday	Cheap	No	Yes
Visit3	Sam's	Lunch	Saturday	Cheap	No	No
Visit4	Kim's	Breakfast	Sunday	Cheap	Yes	No
Visit5	Sam's	Breakfast	Sunday	Expensive	Yes	No
Visit6	Kim's	Lunch	Saturday	Cheap	No	Yes
Visit7	Kim's	Lunch	Monday	Expensive	No	No



19.6 Identification trees:

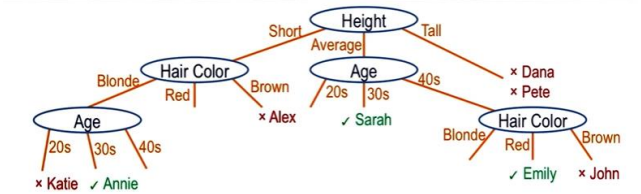
19.6.1 Decision tree learning:

Number	Restaurant	Meal	Day	Cost	Allergic Reaction?
Visit1	Sam's	Breakfast	Friday	Cheap	Yes
Visit2	Kim's	Lunch	Friday	Expensive	No
Visit3	Sam's	Lunch	Saturday	Cheap	Yes
Visit4	Bob's	Breakfast	Sunday	Cheap	No
Visit5	Sam's	Breakfast	Sunday	Expensive	No

19.6.2 Optimal identification tree:

Name	Hair	Height	Age	Lotion	Burn?
Sarah	Blonde	Average	20s	No	Yes
Dana	Blonde	Tall	30s	Yes	No
Alex	Brown	Short	30s	Yes	No
Annie	Blonde	Short	30s	No	Yes
Emily	Red	Average	40s	Yes	Yes
Pete	Brown	Tall	40s	No	No
John	Brown	Average	40s	No	No
Katie	Blonde	Short	20s	Yes	No
Josh	Red	Short	20s	No	Yes

Example – optimality:



19.7 The cognitive connection:

19.7.1 How far to generalize?

19.7.2 Cognitive flexibility – occurs when the agent has multiple characterizations on the same thing; several possible definitions of a concept that converge over time

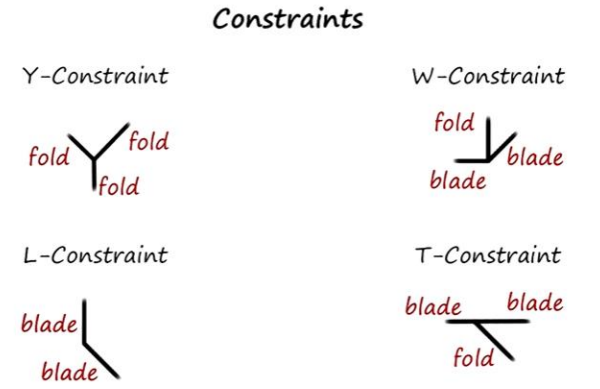
20. Constraint propagation

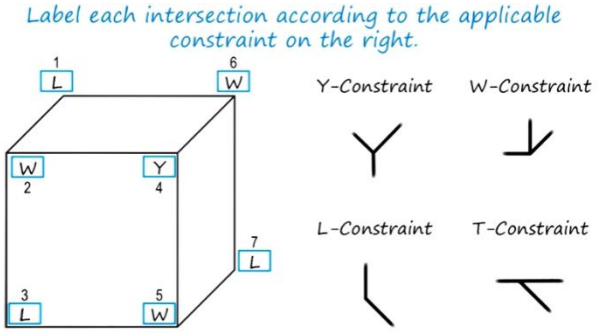
20.1 Constraint propagation is a method of inference that assigns values to variables characterizing a problem in such a way that some conditions (called constraints) are satisfied.

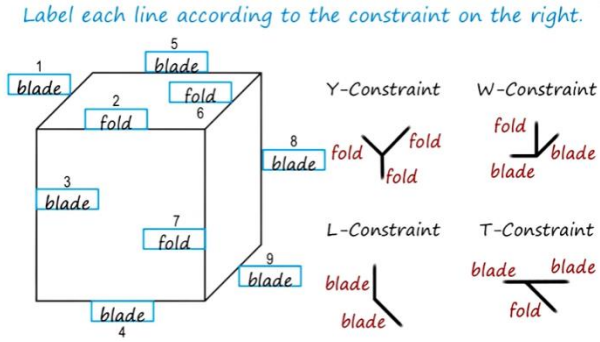
20.2 Pixels to 3D:

- 20.2.1 Lines
- 20.2.2 Surfaces and orientations
- 20.2.3 3D object

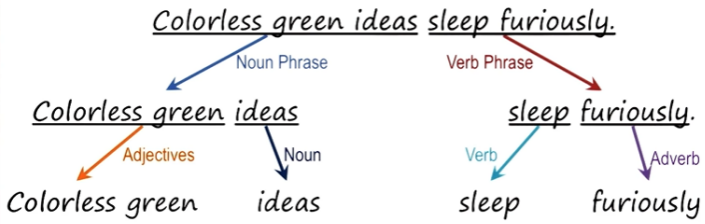
20.3 Constraints intersections and edges (additional complexity possible):







20.4 Constraint propagation for natural language processing:



Constraints:

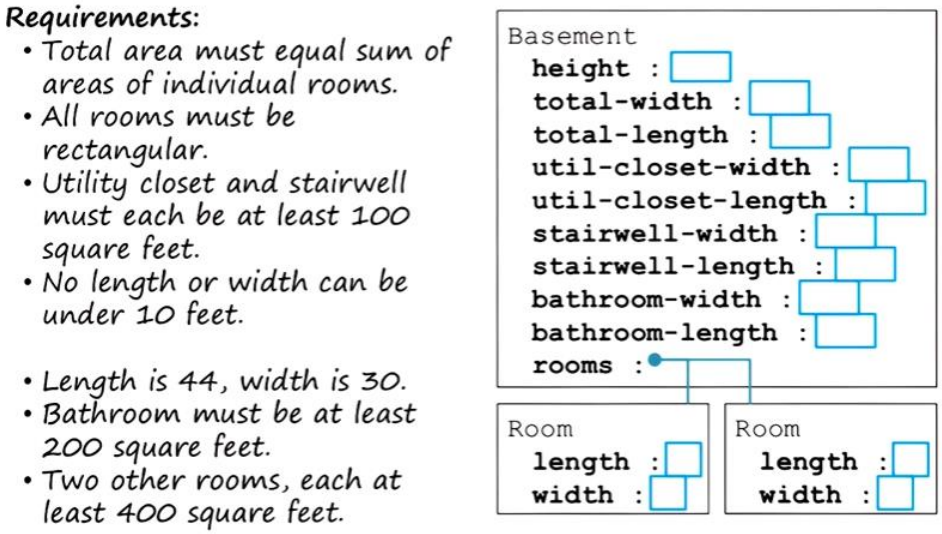
Sentence = Noun Phrase + Verb Phrase

Noun Phrase = [Adjectives] + (Noun or Pronoun)

Verb Phrase = Verb + [Adverb]

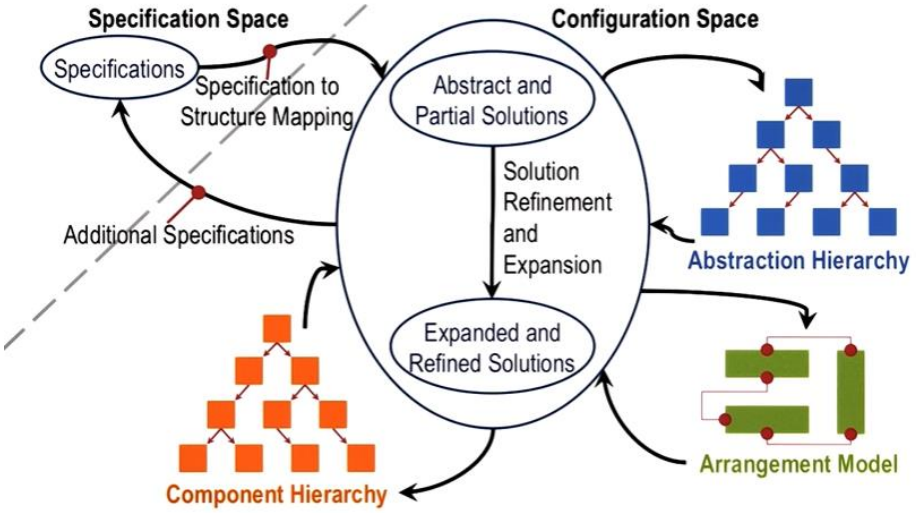
21. Constraint propagation

21.1 Configuration – a very routine kind of a design task in which all the components of the design are already known. The task now is to assign values to the variables of those components so that they can be arranged according to some constraints.

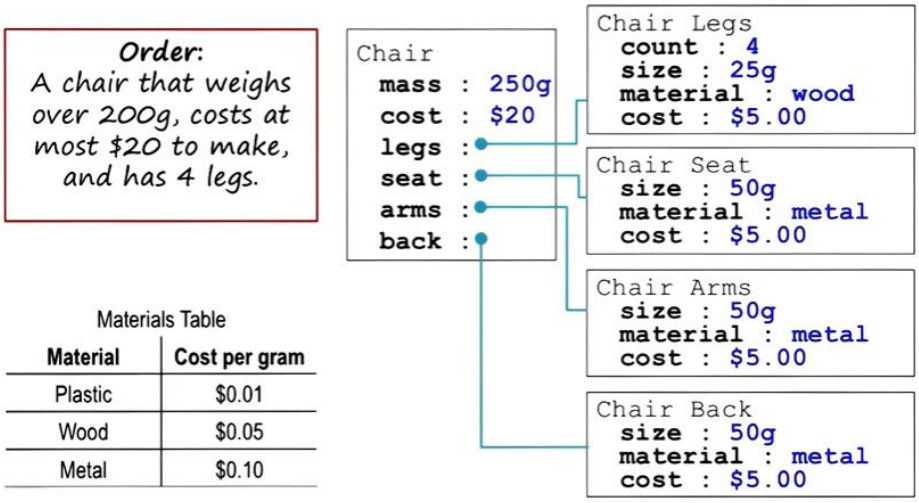


21.2 Design – provides output (some artifact) that satisfies given input (needs, function or goals); wide-ranging, open-ended and well-defined; both the problem and the solution evolve

21.3 Configuration process:



21.4 Applying a constraint:



Materials Table

Material	Cost per gram
Plastic	$0.01
Wood	$0.05
Metal	$0.10

21.5 Connection to classification:

- 21.5.1 Both are hierarchical
- 21.5.2 Configuration (creating the world) leverages classification's (making sense of the world) notion of prototype concepts

21.6 Contrast with case-based reasoning:

- 21.6.1 Configurations suggests starting with a prototype concept and assigning values to variables
- 21.6.2 Case-based reasoning suggests starting from a specific chair and tweaking it as needed

21.7 Connection to planning – the result of planning task can lead to a prototype that subsequently be configured for similar problems with differing constraints

21.8 "Some people even claim that design is a single cognitive activity that has the most economic value of all such activities."

22. Diagnosis

22.1 Diagnosis is the identification of a fault or faults responsible for a malfunctioning system; mapping data space to a hypothesis space

- 22.1.1 Principle of coverage – making sure that the diagnostic conclusion actually accounts for all the input data
- 22.1.2 Principle of parsimony – in general, we want a simple hypothesis but explaining the entire data
- 22.1.3 Explanation – we want a set of hypothesis that would explain the input data

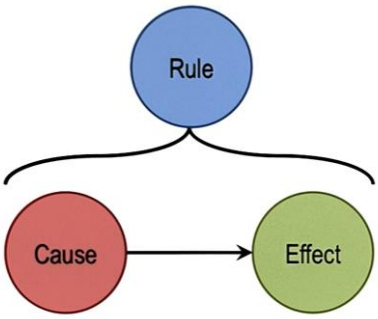
22.2 Diagnostic methods:

- 22.2.1 Rule-based reasoning
- 22.2.2 Case-based reasoning
- 22.2.3 Model-based reasoning

22.3 Problems with diagnosis as classification:

- 22.3.1 One data point, multiple hypotheses
- 22.3.2 One hypothesis, multiple sets of data
- 22.3.3 Multiple hypotheses, multiple sets of data
- 22.3.4 Mutually exclusive hypotheses
- 22.3.5 Interacting data points

22.4 Deduction – induction – abduction

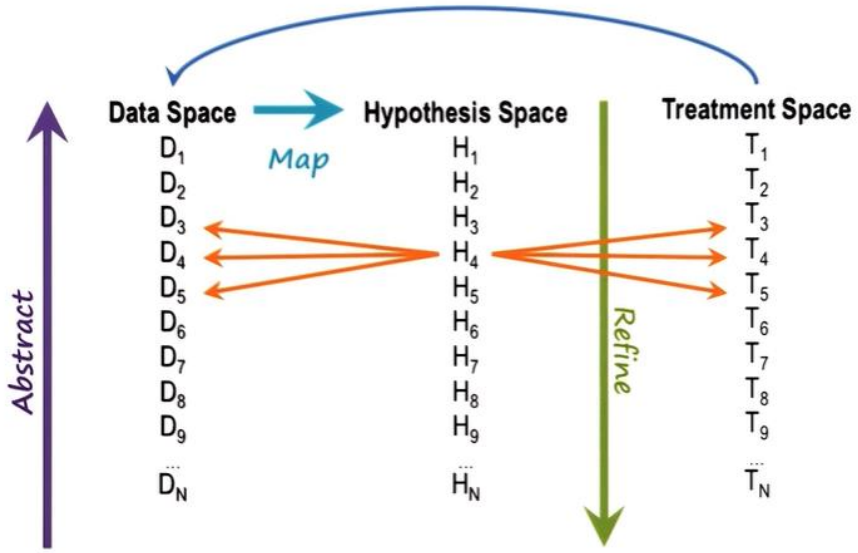


- 22.4.1 Deduction – given the rule and the cause, deduce the effect
- 22.4.2 Induction – given a cause and an effect, induce a rule
- 22.4.3 Abduction – given a rule and an effect, abduce a cause; diagnosis is an instance of abduction

22.5 Criteria for choosing hypotheses:

- 22.5.1 Hypotheses must cover as much of the data as possible (principle of coverage)
- 22.5.2 The smallest number of hypotheses ought to be used (principle of parsimony)
- 22.5.3 Some hypotheses may be more likely than others

22.6 Completing the process:


Data Space
Map
Hypothesis Space
Treatment Space
Abstract
Refine
D1
D2
D3
D4
D5
D6
D7
D8
D9
...
DN
H1
H2
H3
H4
H5
H6
H7
H8
H9
...
HN
T1
T2
T3
T4
T5
T6
T7
T8
T9
...
TN

23. Learning by correcting mistakes

23.1 "Failures are opportunities for learning."

23.2 Questions for learning from mistakes:

- 23.2.1 How can the agent isolate the error in its former model?
- 23.2.2 How can the agent explain the problem that led to the error?
- 23.2.3 How can the agent repair the model to prevent the error from recurring?

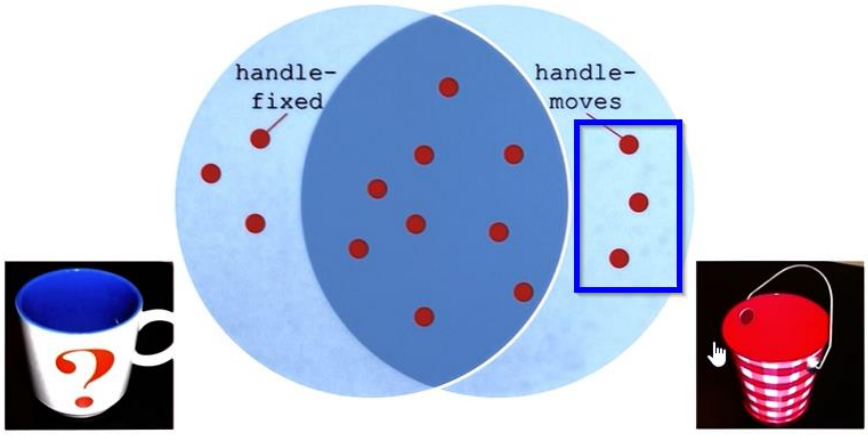
23.3 Credit assignment – problem of identifying the error in one's knowledge that led to failure

23.4 In general, the errors may lie :

- 23.4.1 in the knowledge;
- 23.4.2 in the reasoning;
- 23.4.3 or in the architecture of an agent.

23.5 Focus on errors in knowledge, in particular – errors in classification knowledge

23.6 True suspicious features and false suspicious features:



23.7 Algorithm for isolating mistakes:

Algorithm for Isolating Mistakes

To find suspicious true-success relations:
 Intersect all true successes (∩T)
 Union all false successes (∪F)
 Remove assertions in union from
  intersection (∩T - ∪F)

To find suspicious false-success relations:
 Intersect all false successes (∩F)
 Union all true successes (∪T)
 Remove all assertions in union from
  intersection (∩F - ∪T)

23.8 Explanation-free repair:

Old Rule

```
If:
  Object has bottom
  Bottom is flat
  Object has concavity
  Object is lightweight
  Object has a handle

Then:
  Object is a cup
```

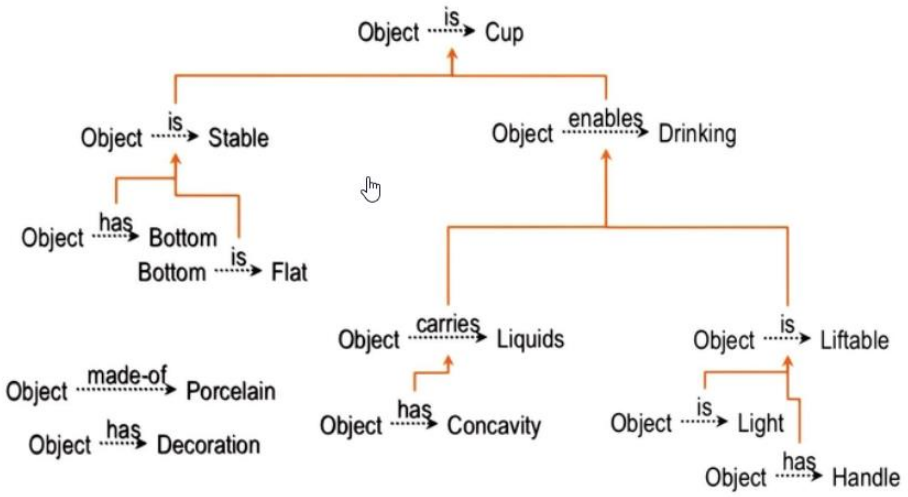
New Rule

```
If:
  Object has bottom
  Bottom is flat
  Object has concavity
  Object is lightweight
  Object has a handle
  Handle is fixed

Then:
  Object is a cup
```

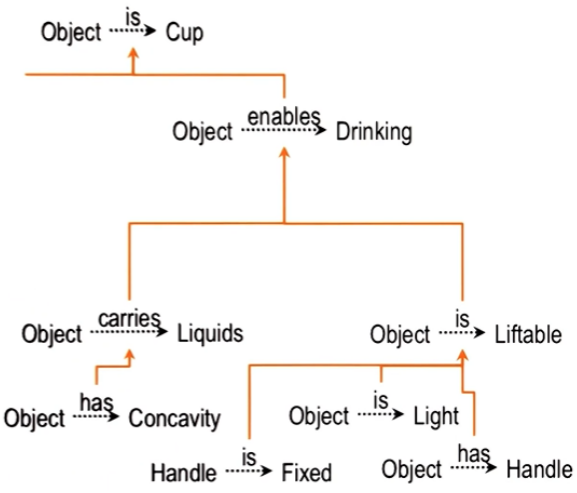
23.9 "Classification is ubiquitous in many schools of AI. (…) Explanation, however, is a key characteristic of knowledge-based AI. It leads to deeper learning."

23.10 Explaining the mistake:

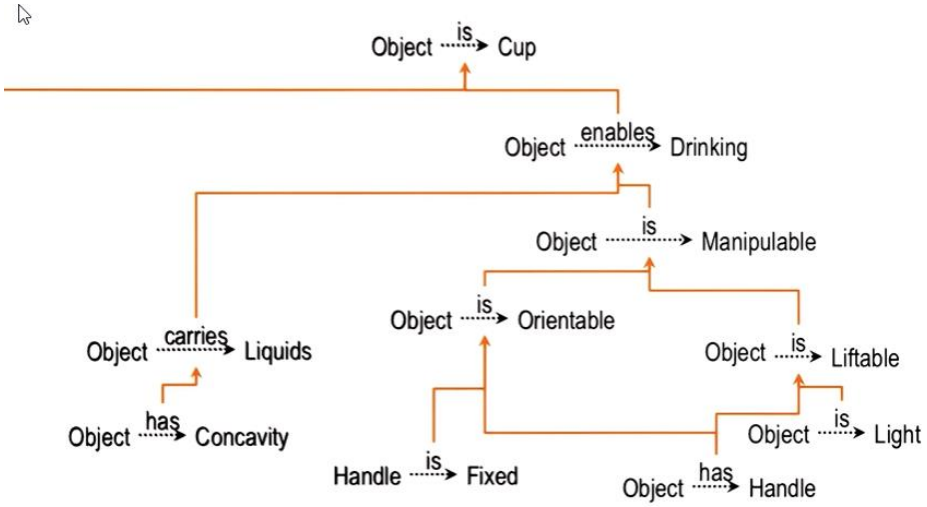


Is this a good way to fix this error?

- Yes, because it shows only fixed-handle cups enable drinking.
- No, because it will exclude some actual cups.
- No, because some non-cups will still be included.
- ● No, because it will cause incorrect decisions about other objects.



23.11 Correcting the mistake from 23.10:

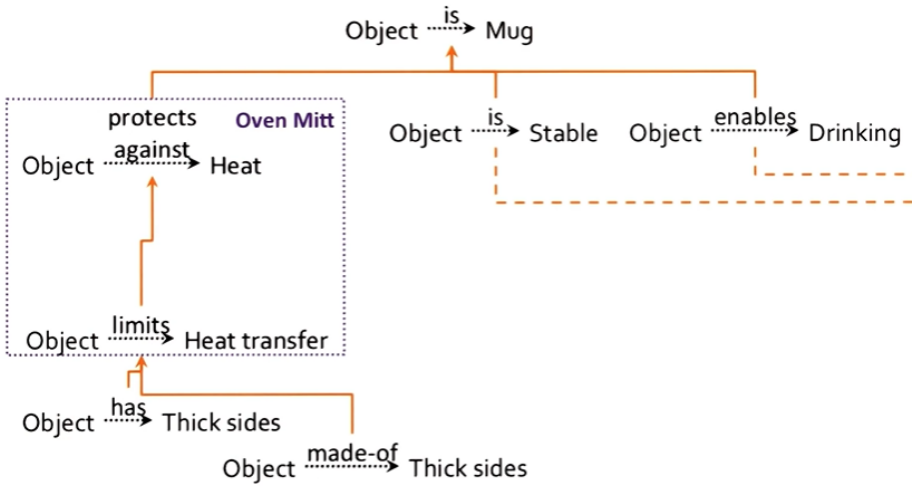


23.12 "Learning by correcting mistakes uses learning as a problem-solving activity. (…) This learning is closely intertwined with memory, reasoning, action, and feedback from the world."

24. Meta-reasoning

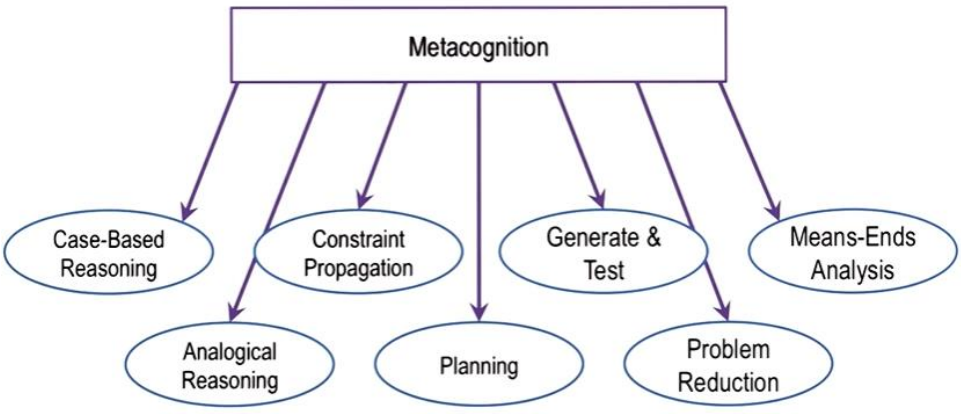
24.1 Meta-reasoning – "thinking about thinking"; "how do you know that you don't know"

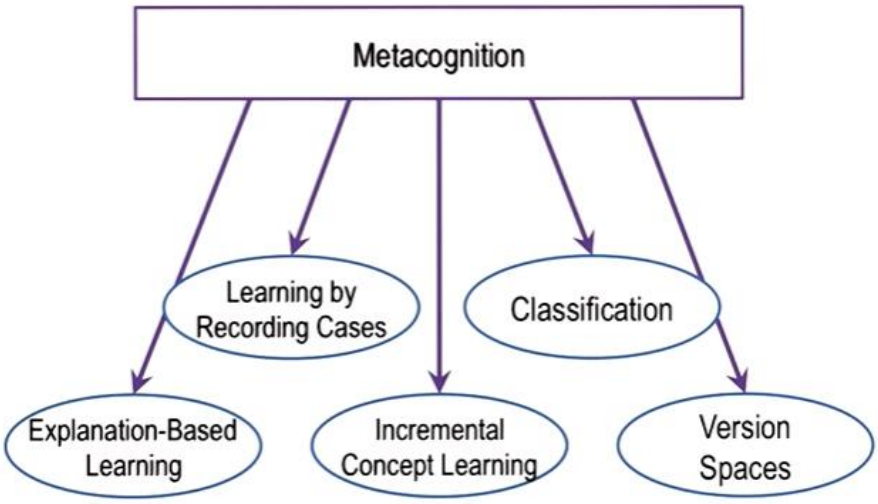
24.2 Knowledge gaps (cases in which it is incomplete) lead to setting learning goals (connecting the dots)



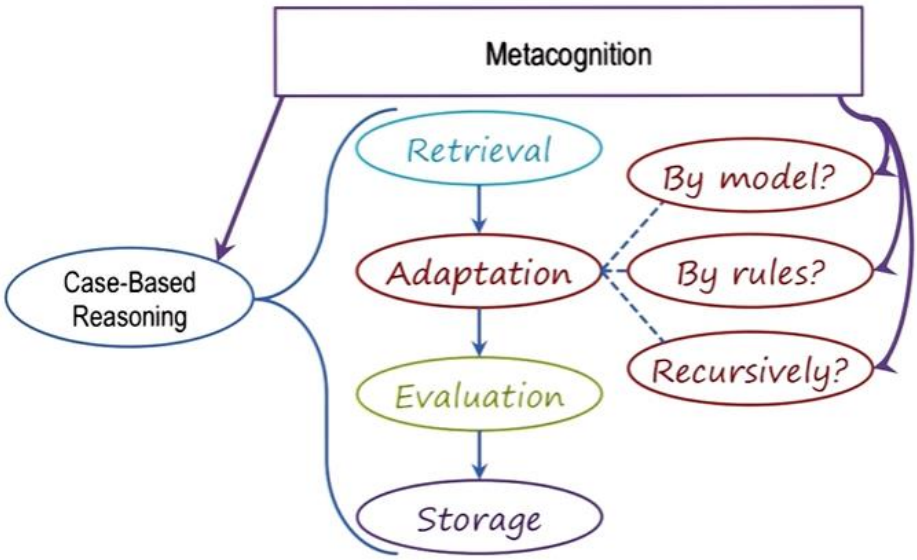
24.3 "There might be considerable overlap between metacognition and deliberation."

24.4 Strategy selection – metacognition will select one particular method depending on what knowledge is exactly available for addressing the specific input problem; it might also select between computing methods, for instance, based on their computational efficiency or the quality of solutions (logic)





24.5 Strategy integration:



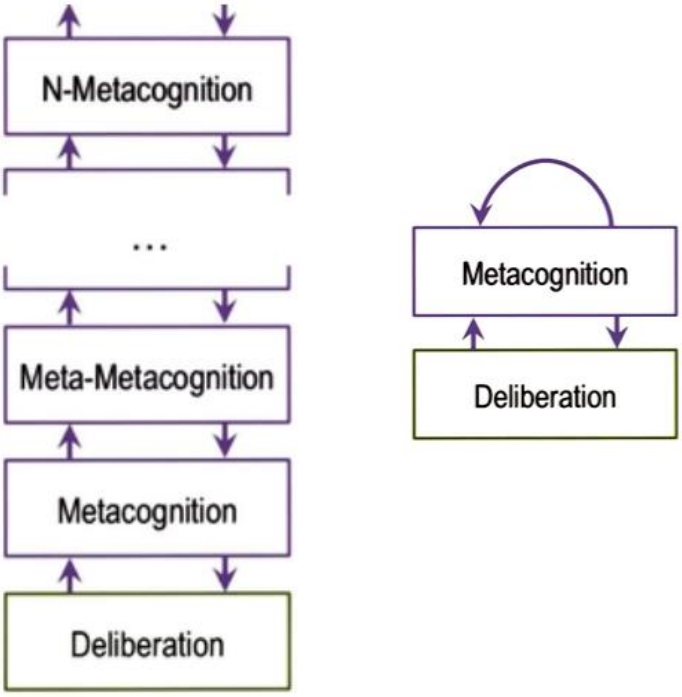
24.6 Process of meta-reasoning can be used:

24.6.1 to fix gaps in knowledge reasoning or learning;

24.6.2 for strategy integration;

24.6.3 etc.

24.7 N-metacognition model (left) is not a good way to think about levels of metacognition "because each level of metacognition is conceptually identical, so they are better represented as self-referential (right)."



24.8 Goal-based autonomy – we want agents that can adapt their reasoning methods and their learning methods to try to achieve the new goal, even if they were not necessarily programmed to achieve that goal

25. Advanced topics

25.1 Visuospatial knowledge – knowledge wherein causality is, at most, implicit

25.2 Two views of reasoning:

- 25.2.1 working on propositional representations;
- 25.2.2 working on analogical representations.

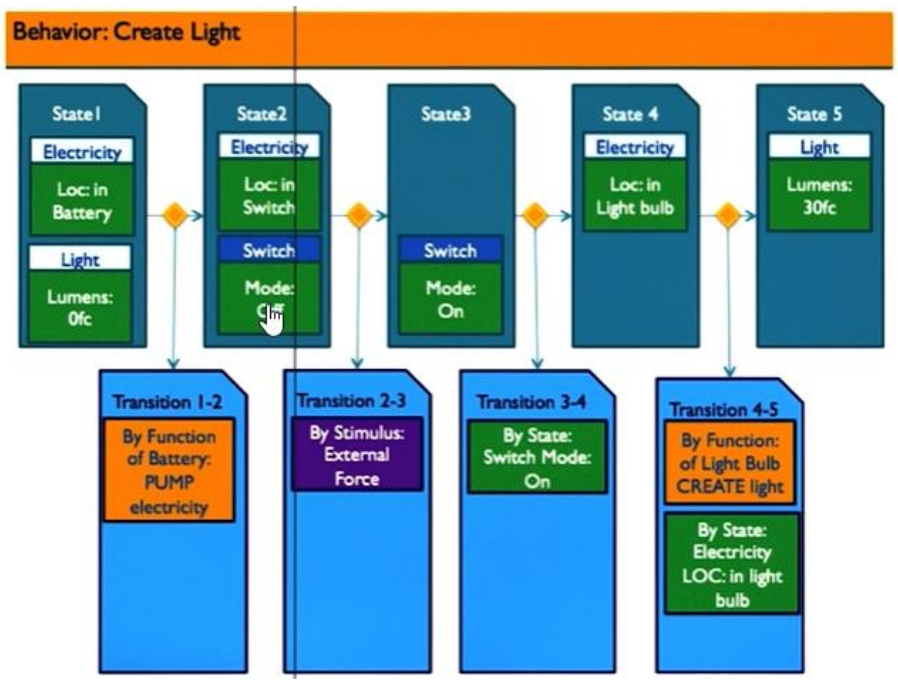
25.3 Symbol grounding problem:

	Visuospatial	Verbal
Content	Appearance: What and Where	Arbitrary: Driven by Inferential Needs
Encoding	Analogical: Structural Correspondence	Propositional: No Correspondence

25.4 Systems thinking – reasoning about systems with numerous components and processes at multiple, potentially invisible, levels of abstraction. Connections to:

- 25.4.1 frames;
- 25.4.2 scripts;
- 25.4.3 diagnosis.

25.5 Structure-Behavior-Function model:



25.6 Design-thinking – reasoning about ill-defined, unconstrained, open problems that are situated in the world, e.g. sustainability. Agents doing design:

- 25.6.1 Configuration design
- 25.6.2 Case-based reasoning

25.7 "Every time you create a design, every experience is an opportunity for learning."

25.8 Creativity – anything that produces "a non-obvious, desirable product"

25.9 Something is creative if it is:

- 25.9.1 Novel (newness)
- 25.9.2 Valuable
- 25.9.3 Unexpected (something non-obvious or surprising)

25.10 Some other processes of creativity:

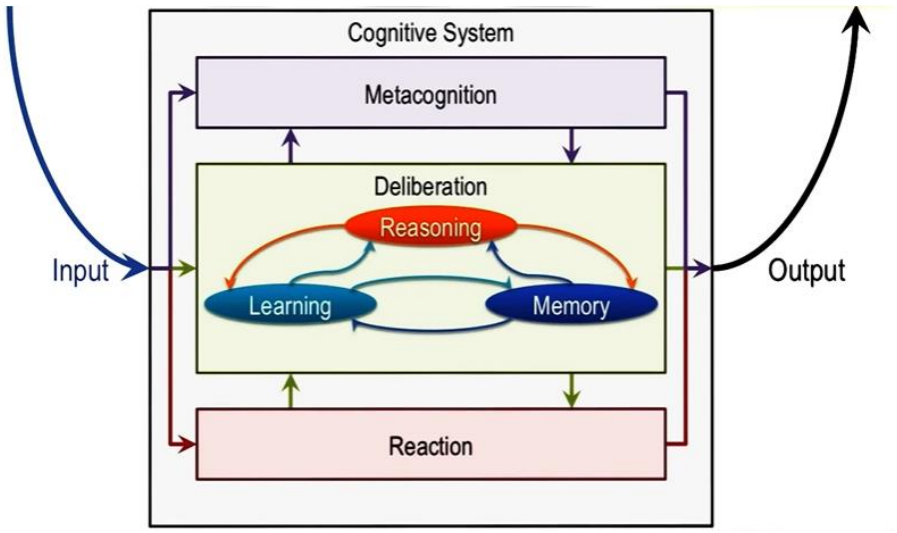
- 25.10.1 Emergence
- 25.10.2 Re-representation – original representation of a problem is not conducive to problem solving)
- 25.10.3 Serendipity – "One kind of serendipity occurs when I'm trying to address a problem and I am unable to address it, so I suspend the goal and I'd start doing something different. Later, at some other time, I come across a solution and connect it with the previous suspended goal."

25.11 AI Ethics – "are we doing the right kind of things?"

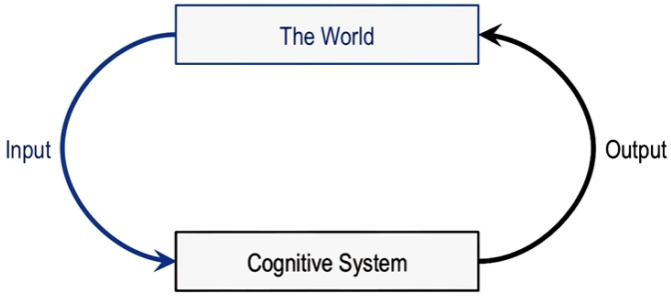
- 25.11.1 Human economy and society
- 25.11.2 Robot soldiers and morality
- 25.11.3 Civil rights for robots

26. Wrap-up

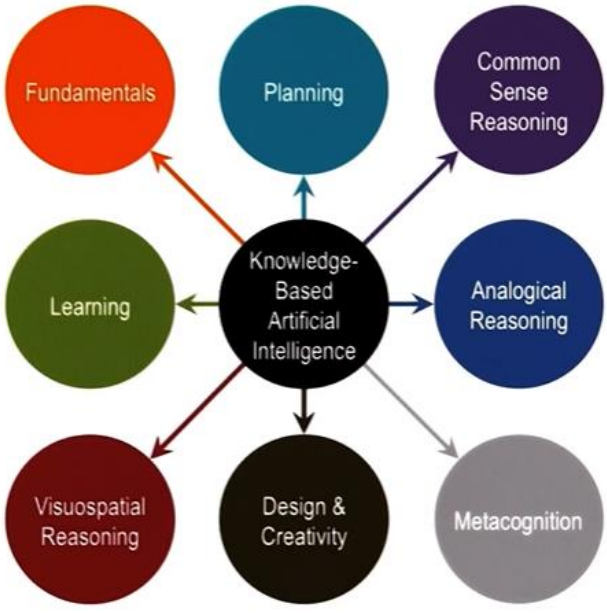
26.1 Cognitive systems – overlapping spaces:



26.2 Constant flow:



26.3 Course structure revisited:



26.4 Principle #4:

Principle #4: KBAI agents match methods to tasks.

Methods	Tasks
Generate & Test	Configuration
Means-Ends Analysis	Diagnosis
Problem Reduction	Design
Production Systems	Meta-Reasoning
Case-Based Reasoning	Creativity
Planning	Classification
Analogical Reasoning	Systems Thinking

26.5 Current research:

CALO:
"Cognitive Assistant that Learns and Organizes"

Cyc and OMCS:
Knowledgebases of everyday common sense knowledge

Wolfram Alpha:
A computational knowledge and answer engine.

VITA:
A computational model of visual thinking in autism, based on RPM.

Dramatis:
A computational model of suspense and drama in stories.

DANE:
Support for design based on analogies to natural systems.

This document was created by me (Sylwia Maria Rapacz) but note that **all of the material above is based on the lecture modules of the Knowledge-Based AI course (CS7637) and was prepared by the course instructors – David A. Joyner, PhD and Professor Ashok K. Goel**.