Module 1: intro to knowledge based AI

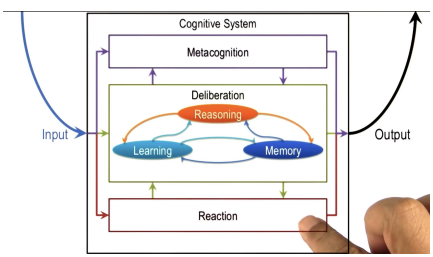
Characteristics of AI problems

- Knowledge often arrives incrementally
- problems exhibit recurring patterns
- problems have multiple levels of granularity
- many problems are computationally intractable
- the world is dynamic, but knowledge of the world is static
- the world is open- ended, but knowledge is limited

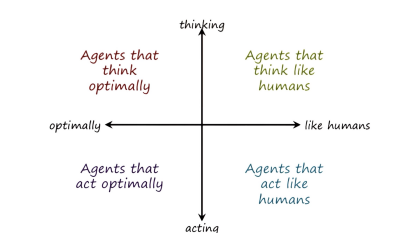
Characteristics of AI agents

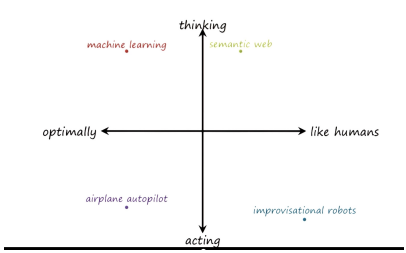
- agents have limited computing power
- agents have limited sensors
- agents have limited attention
- computational logic is fundamentally deductive
- AI agents' knowledge is incomplete relative to the world

Knowledge based AI



Four schools of AI





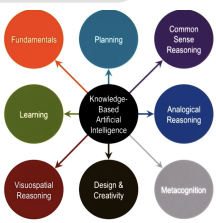
What are cognitive systems

Cognitive: dealing with human-like intelligence

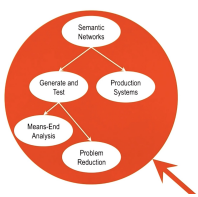
Systems: multiple interacting components such as learning, reasoning, and memory

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

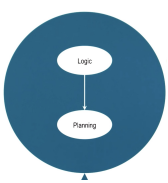
Topics In kbai



Fundamentals

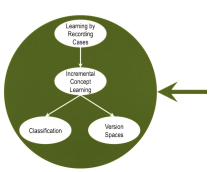


Planning

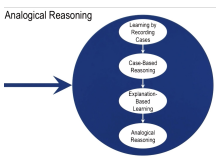


Common Sense Reasoning

Learning

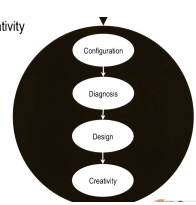


Analogical Reasoning

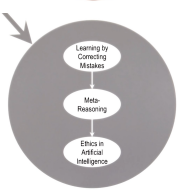


Visuospatial Reasoning

Design & Creativity



Metacognition



Module 2: Computational Psychometrics

Computational Psychometrics

- psychometric = study of human intelligence, aptitude, knowledge
- computational Psychometrics is the design of computational agents that can take the same tests that humans do when they are tested for intelligence or knowledge or aptitude

Raven's progressive matrices

- Test of intelligence
- strictly visual questions
- 2x2 matrix problems
- 3x3 matrix problems
- 2x1 matrix problems

Module 3: Semantic Networks

Common sense reasoning

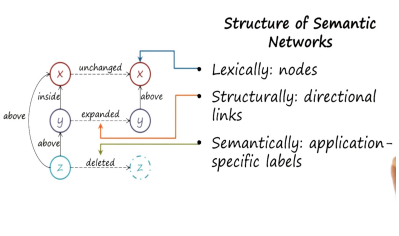
- Language, vocabulary, content, knowledge
- Start by representing objects
- then represent the relationships of these objects (between them)
- common sense reasoning - based an common sense knowledge you can make inferences about the perceived world around you
- just because you didn't directly observe the rain you can make inference because you saw someone using an umbrella - common sense reasoning
- when you are able to make logical inferences based on everyday knowledge and context

Semantic networks

- Type of knowledge representation
- provide visual representations of relationships between different concepts or nodes
- used for problem solving and reasoning
- by representing knowledge in a structured way they enable us to apply reasoning algorithms to draw conclusions or make inferences based on the relationships in the network
- have a connection to human cognition
- human mind is similar to semantic networks in the same way problems and knowledge are represented using a similar network like structure
- we are accessing knowledge stared in our minds

Humans and semantic networks

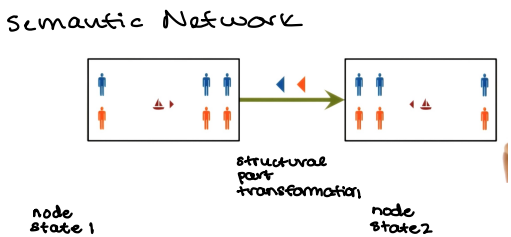
- Humans have semantic net works
- human mind naturally organizes and represents knowledge using a network like structure
- dog - animal, pet, bark
- store and retrieve information, make associations, reason about the relationships between different concepts
- used to help us organize and connect information

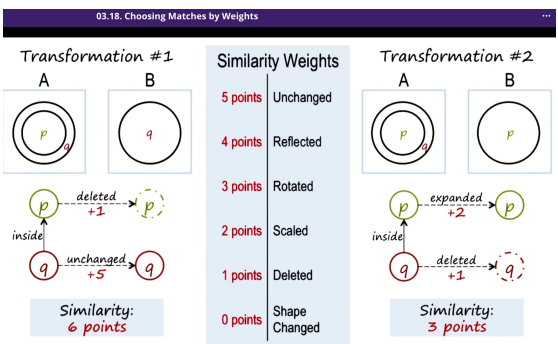


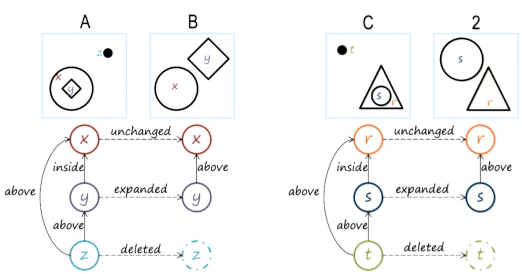
Guards and prisoners problem

Guards & Prisoners Problem

- Three guards and three prisoners must cross 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).





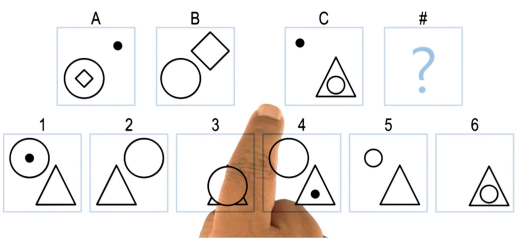


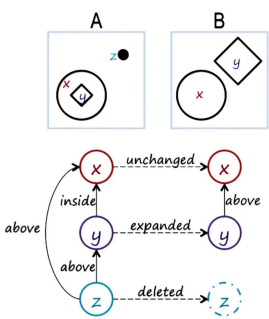
Is this the right answer to the problem? ● Yes ○ No

Module 4: Generate and Test

Problem solving

- Tester generates all possible states and removes some
- combinatorial explosion _ start with a small number of states but the number of successor states keeps increasing very rapidly
- this happens with a dumb tester and generator
- generated tests are a powerful problem solver
- semantic networks need a problem solving method that uses knowledge representation to actually solve the problem
- generating test is a problem solving method
-





To recap...

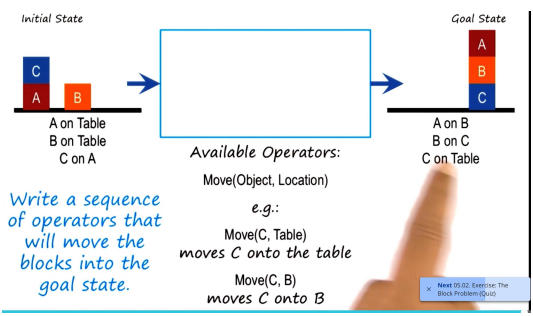
- Generate and test
- Smart testers
- Smart generators
- Generate and test in an unconstrained domain

Module 5: means-ends analysis and problem reduction

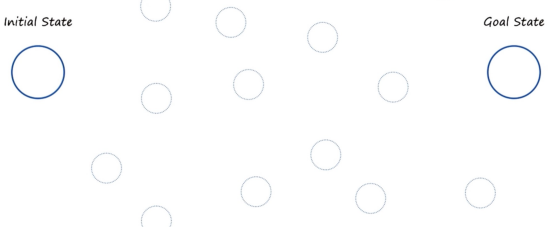
State spaces

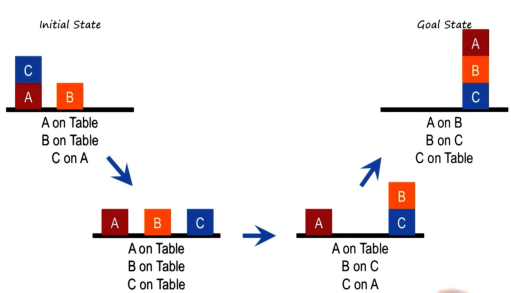
Lesson Preview

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



State Space





Differences in state space

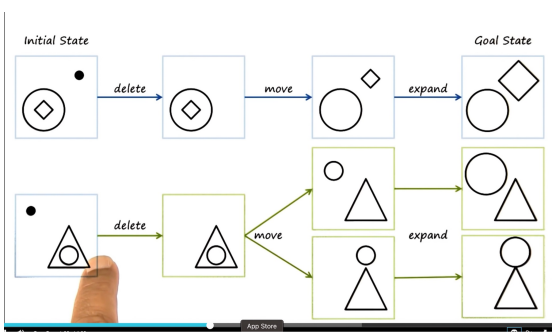
- The difference between different state and goal state - end
- The application of the operator is the means - means ends analysis
- at any state I am going to pick an operator that will reduce the distance between the current state and goal state \

Means-Ends Analysis

For each operator that can be applied:

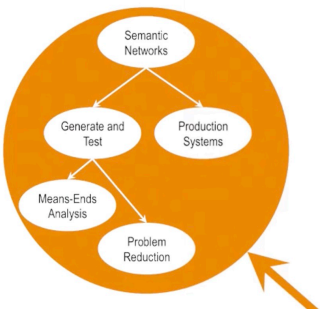
- Apply the operator to the current state
- Calculate difference between new state and goal state

Prefer state that minimizes distance between new state and goal state

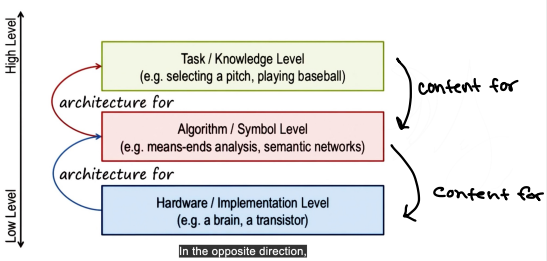


Module 6: Production Systems

Fundamentals



Levels of cognitive architecture



What are the layers of Watson?

the physical computer
searching and decision-making
answering the inputted clue

Assumptions of a cognitive architecture

Assumptions of a Cognitive Architecture

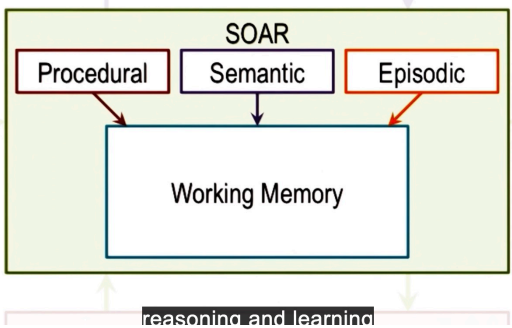
- Goal-oriented
- Rich, complex environment
- Significant knowledge
- Symbols and abstractions
- Flexible and function of the environment
- Learning

- Function for cognitive architectures:

$f : P^* \rightarrow A$

Percepts → Action

long term memory structure



reasoning and learning

- Chunking - soar has a possible rules - it has an impasse
- might there be away to learn a rule to decide between the 2 rules? It will use its episodic knowledge - becomes a procedural rule - chunking is a learning method

Module 7: Frames

Slots and Fillers

- frames are a knowledge structure

Ashok ate a frog.

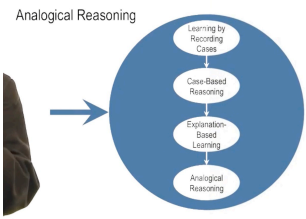
```
Ate
    subject : Ashok
    object : a frog
    location :
    time :
    utensils :
    object-alive : false
    object-is : in-subject
    subject-mood : happy
```

David ate a pizza at home.

```
Ate
    subject : David
    object : a pizza
    location : at home
    time :
    utensils :
    object-alive : false
    object-is : in-subject
    subject-mood : happy
```

Module 8: Learning by Recording Cases

Learning

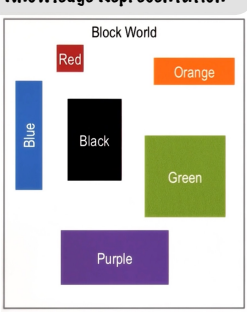


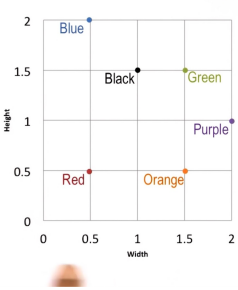
Lesson Preview

Given problem a
Retrieve most similar prior problem, b, from memory
Apply b's solution to problem a

- Learning by recording cases
- Nearest neighbor method
- k-Nearest Neighbor
- Cases in the real world

Knowledge Representation





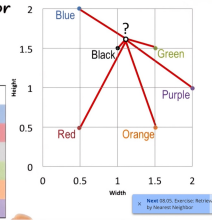
Nearest Neighbor

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



Euclidian distance

ex. $x_n, y_n = 0.8, 0.8$

Blue = 0.5, 2.0

$d = \sqrt{(2.0 - 0.8)^2 + (0.5 - 0.8)^2}$

$= \sqrt{1.44 + 0.9}$

$= 1.529$

Red = 0.5, 0.5

$= 0.42$

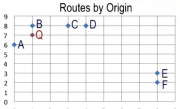
Finding the Nearest Neighbor

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

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

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

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



- **consistency heuristic:** whenever you want to guess a property of something given nothing else to go on but a set of reference cases and the most similar case as measured by known properties for which the property is known. Guess that the unknown property is the same as that known property
- ex. Plotting widths and heights of the cases → makes it easy to apply the consistency heuristic
- nearest neighbors → calculate the distance to each other block and then find the minimum
- best way to do this is a decision tree (semantic tree) → divide up cases in advance of nearest-neighbor calculation
- ex. Divide by height, ten divide each set by width
- k-d tree: decision tree
- A) set of possible answers consists of points one of which maybe the closest neighbor
- B) each test specifies a coordinate, a threshold and a neutral zone around the threshold containing no points
- c) each test divides a set of points into two sets according to which side of the threshold each point lies

To find the nearest neighbor using the K-D procedure,

- Determine whether there is only one element in the set under consideration.
  - If there is only one, report it.
  - Otherwise, compare the unknown, in the axis of comparison, against the current node's threshold. The result determines the likely set.
  - Find the nearest neighbor in the likely set using this procedure.
  - Determine whether the distance to the nearest neighbor in the likely set is less than or equal to the distance to the other set's boundary in the axis of comparison:
    - If it is, then report the nearest neighbor in the likely set.
    - If it is not, check the unlikely set using this procedure; return the nearer of the nearest neighbors in the likely set and in the unlikely set.

Module 9: Case Based Reasoning

Lesson Preview

- Need for case-based reasoning
- Case adaptation, evaluation, and storage
- Case retrieval revisited
- Advanced case-based reasoning

- **Case-based reasoning:** the cognitive agent addresses new problems by tweaking solutions to similar previously encountered problems
- builds upon **learning recording cases** → where the new problem is identical to the previous problem
- in **case-based reasoning** the problems are only similar and has several phases, **case retrieval, case adaptation, case evaluation** and case storage
- we can extract something from memory but teen need to apply reasoning to it

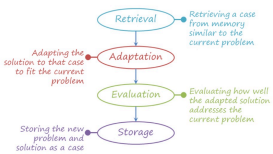


Figure 261: Recording Cases to Case-Based Reasoning

Case- based reasoning

- **retrieval:** k-nearest neighbor is one way of retrieving cases from memory
- once we retrieved a case from memory that is delivered to the current problem - need to adapt it
- **adapt:** once we adapt the case to meet the requirements of the new problem we have a candidate solution for the new problem
- **Evaluate:** possibly via simulation
- **Storage:** we encapsulate the new problem and the new solution into a case and stone the case back into memory
- this process unifies memory, reasoning and learning
- Reason when we adapt and evaluate learn wen western
- assumptions: there are patterns to the problems that agents encounter **in the ward, similar** problems often have similar solutions

Case - adaption

- 3 most common ways of adapting a case: **model-based method, recursive case-based method** and the **rule-based method**
- **model-based method:** retrieve a similar model, do a search using this case, adapt the earlier case using some model of reward
- **case adaptation by recursive reasoning:** don't know how to go from home to restaurant but I do know how to go from my office to a restaurant - use case of going from home to office and the retrieved case from office to restaurant - combine to get a solution
- **case based reasoning** the first time you retrieve a case for solving a problemthe case provides a partial solution, take the remaining part that was not solved and make it a new problem and send it back into case memory, case memory then ends another case - combine cases to get a full solution
- solve problem by recursively breaking a problem down into smaller problems
- case-adaption by rules: method uses heuristics expressed in the form of **rules**
- heuristics is a rule of thumb, works often but not always, ex. You look around **and you and where the tallest** building is
- if we want to go back home from the restaurant, a heuristic might say to just Crip all the terms from the previous case

Case-evaluation

- Evaluation step is concerted with auto assess the suitability of the candidate solution to the problem
- simulation → an evaluation method, case based reasoning proposed a candidate solution, simulation contest ifit actually works
- need to consider cost of executing a solution - simulating can save us this cost and we are evaluating whether or not our adaptation successfully solved our new problem, when itdoesn't we go back to the adaptation phase and try again or go to the retrieval phase to get anchor case to use to inform our solution

Case storage

- Once we have the new problem and solution for it we can encapsulate them as a case and stare them in memory - constantly accumulating and assimilating new cases
- **storage methods: indexing** and **discrimination tree**
- **indexing:** could include indexes such as whether a route is scenic or not or fast or not fast then each of those values then became a particular way of identifying each individual case such that when given a new problem, I can find the most similar case by seeing which one matches the most of throes variables
- indexical structure allows for effective and efficient retrieval because we are storing things only because we want to retrieve them at a later time.

Figure 285: Case Storage by Discrimination Tree

- A **descrimination tree** is a knowledge structure in which the cases themselves are the leaf node of the tree.
- root node and at all the parent nodes are questions (intimidated node)
- **intimidated node** pertains to the indexical structures of the cases - using the origins of the cases as the index equals structure
- How will we incrementally learn this knowledge structure as new cases are put into the case library? We must find a way of discriminating between A and X so we will add a new question here
- each time we add a location to memory the organization of the case of memory changes - example of incremental learning - with the addition of each new case some new **knowledge structure is learned**
- by asking a question we can prune away are part of the tree
- both the table and discrimination tree are trying to accommodate and **accumulate new cases**
- bigo-logarithmic, for efficiency of searching the case library organized by **indices = linear**
- **sometimes studying** failed cases is good
- can try to repair a **failure by going** back to evaluation step - not always useful to store every failed or successful case
- k-nn → used to make a weighted sum of each case feature similarity **and form** a retrieved case set which includes a certain number of cases

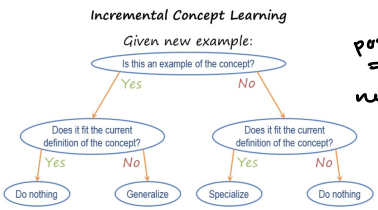
Module 10: incremental concept learning

Lesson Preview

- Purpose of incremental concept learning
- Variabilization
- Specialization
- Generalization
- Heuristics for specialization and generalization

Figure 305: Preview

- Background information is important



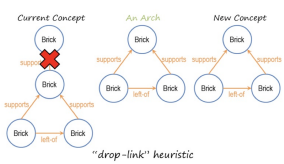
positive example = generalize

negative example = specialize

Variabilization

- AI program can only variables - brick A is an instance of brick therefore I just have a brick here

Generalization to ignore features



generalized so current concept can cover new concept

Drop-link heuristic

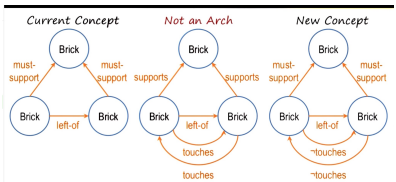
- use if you need to generalize in such a way that the new concept can cover both earlier examples
- useful when the structure of the covert and new example have alot of overlap

Require - link heuristic

- rule out negative example, we will put extra conditions on those links
- if the structure of the representation of the concept and is structure of the representation of the negative example have some things in common but there are also some differences
- those things that ave not in common be come must be required

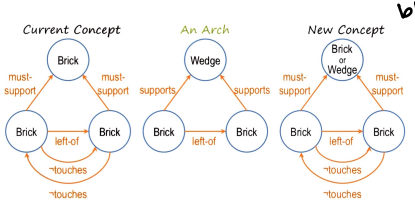
add a link - revise current concept so that things that are not required are required to cover both concepts

Specialization to exclude features



negate something in new concept to help it match current concept

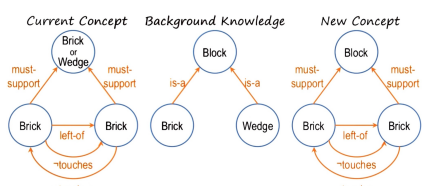
Generalization to abstract features



enlarge set by combining two concepts to create a new generalized concept

Climb-tree heuristic

- use background informations to generalize the new concept so that i.e. A wedge and brick are both blocks



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

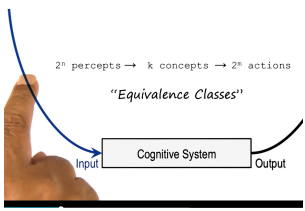
close-interval: expand range of values to be a positive example of the concept

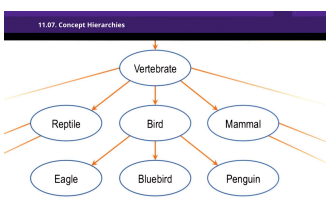
Module 11: Classification

Lesson Preview

- Equivalence classes and hierarchies
- Kinds of concepts
- Bottom-up and top-down processes

- Cognitive system with inputs and outputs processes classification
- if you have too many precepts then there may be too many combinations and that becomes the challenge of classification





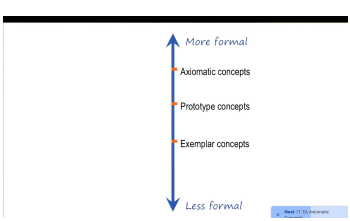
super class vs. subclass

How would you characterize the class 'bird' given the characterization of its subclasses below?

Bird

Lays eggs?	Has talons?	Has fur?
● Yes ○ No ○ Maybe	○ Yes ○ No ● Maybe	○ Yes ● No ○ Maybe
Has wings?	**Flies?**	**Large?**
● Yes ○ No ○ Maybe	○ Yes ○ No ● Maybe	○ Yes ○ No ● Maybe

Eagle		Bluebird		Penguin	
Lays eggs?	**Yes**	Lays eggs?	**Yes**	Lays eggs?	**Yes**
Has wings?	**Yes**	Has wings?	**Yes**	Has wings?	**Yes**
Has talons?	**Yes**	Has talons?	**Yes**	Has talons?	**No**
Flies?	**Yes**	Flies?	**Yes**	Flies?	**No**
Has fur?	**No**	Has fur?	**No**	Has fur?	**No**
Large?	**Yes**	Large?	**No**	Large?	**Yes**



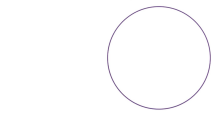
easier to train an agent in

↓

harder

Axiomatic concepts

Axiomatic concepts:
Concepts defined by a formal set of necessary and sufficient conditions.



Circle: all points in a plane that are equidistant from a single point.

Prototype concepts

Prototype concepts:
Base concepts defined by a typical example with overridable properties.

```
Chair
    #-of-legs : four
    material : metal
    has-back : true
    has-arms : false
    is-cushioned : false
    ...
```

Prototype concepts:
Base concepts defined by a typical example with overridable properties.

```
Chair
    #-of-legs : four
    material : metal
    has-back : true
    has-arms : false
    is-cushioned : false
    ...
```

```
Stool
    has-back : false
    ...
```

```
Folding chair
    has-back : true
    #-of-legs : four
    has-arms : false
    ...
```

Exemplar concepts

Exemplar concepts:
Concepts defined by implicit abstractions of instances, or exemplars, of the concept.

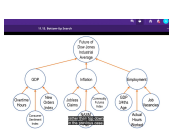
Example: beauty

Order of concepts

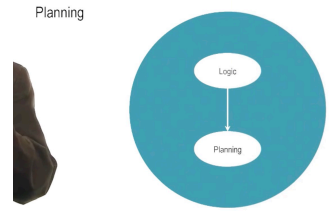
Less formal concept than exemplar concepts such as in philosophy quaila which are raw sensations we get from our sensors such as bitterness

Rank the following concepts based on their formality:

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



Module 12: Logic



Lesson Preview

- Formal notation
- Conjunctions, disjunctions, negations, implications
- Truth tables
- Rules of inference
- Resolution theorem proving

Schemes of logic

Predicate:
A function that maps object arguments to true or false values

predicate object
`Feathers(animal)`

If an animal has feathers, then it is a bird

predicate if-true then
`If Feathers(animal):`
`Then Bird(animal)`
predicate
(object relationship)
(implicative relationship)

If an animal lays eggs and it flies, then it is a bird

`Lays-eggs(animal)`
`∧` → conjunction
`Flies(animal)`

AND
`If Lays-eggs(animal) ∧` conjunction
`Flies(animal):` → implication (then)
`Then Bird(animal)`

If an animal lays eggs or it flies, then it is a bird

`Lays-eggs(animal)`
`∨` OR
`Flies(animal)`

`If Lays-eggs(animal)`
`Flies(animal):` ∨ disjunction
`Then Bird(animal)`

If an animal flies and is not a bird, it is a bat.

`Flies(animal)`
`∧`
`¬Bird(animal)`

`If Flies(animal)` ∧ conjunction
`¬Bird(animal):` implies (then)
`Then Bat(Animal)`
negation

`Lays-eggs(animal) ∧ Flies(animal) ⇒ Bird(animal)`
↗ implication

left hand side implies right hand side

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

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?

Truth tables

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

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

Truth of Implications

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

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

Commutative Property same

A	B	A ∧ B	B ∧ A
True	True	True	True
True	False	False	False
False	True	False	False
False	False	False	False

Distributive Property

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

A AND → when A is false, everything is false
A AND (true) → ignore A and focus on B or C when either B or C are true, True when both B or C are false, false
If an if-true sentence starts in an OR in the middle, then everything is True

sociative means I can change parens

de Morgan's Law

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

Truth of Implications

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

Associative Property

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

Implication Elimination

Given: a ⇒ b
Rewrite as: ¬a ∨ b

Given: Feathers ⇒ Bird
Rewrite as: ¬Feathers ∨ Bird

Rules of Inference: Instantiate general rules to prove specific claims.

Modus Ponens
Sentence 1: p ⇒ q
Sentence 2: p
∴ Sentence 3: q

Feathers ⇒ Bird
Feathers
∴ Bird

Modus Tollens
Sentence 1: p ⇒ q
Sentence 2: ¬q
∴ Sentence 3: ¬p

Feathers ⇒ Bird
¬Bird
∴ ¬Feathers

Module 18: Analogical Reasoning

Lesson Preview

- Similarity and case-based reasoning
- Process of analogical reasoning
- Design by analogy

Cross domain analogy

- Objects and features and the values of the objects can be different the similarity is based on the relationship, relationship gets transferred

More similar

	Relations?	Objects?	Features?	Values?
e.g. Recording Cases	Similar	Similar	Similar	Similar
e.g. Case-Based Reasoning	Similar	Similar	Similar	Dissimilar
	Similar	Similar	Dissimilar	Dissimilar
e.g. Analogical Reasoning	Similar	Dissimilar	Dissimilar	Dissimilar

Less similar

Superficial Similarity

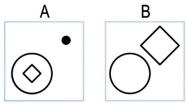
- Features
- Counts
- Objects

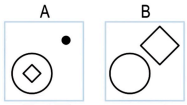
(a unary relationship)

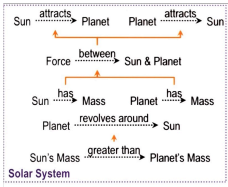
Deep Similarity

(binary relationship)

- Relationships between objects
- Relationships between relationships (high order relationship (ternary))







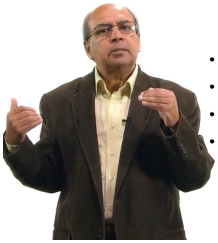
Types of Similarity

Semantic
Conceptual similarity between the target problem and the source case.

Pragmatic
Similarity of external factors, such as goals.

Structural
Similarity between representational structures.

Module 13: Planning



Lesson Preview

- *States, goals, and operators*
- *Conflicts in planning*
- *Partial-order planning*
- *Hierarchical task networks*

Propositional Logic

- **Goal: get the ceiling painted and the ladder is painted**

How would we represent the second part of the goal state?

How would we represent the goal state as a conjunction?

In propositional logic:
Painted (Ceiling)
Painted (Ladder)

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

Name	Symbol	Connection	Meaning
AND (Conjunction)	∧	P ∧ Q	(P ∧ Q) is true if both P and Q are true otherwise false.
OR (Disjunction)	∨	P ∨ Q	(P ∨ Q) is true if either P or Q is true (or both) otherwise false.
NOT (Negation)	¬	¬P	¬P is the opposite of P. If P is true, ¬P will be false and vice versa.
Exclusive OR	⊕	P ⊕ Q	Either P or Q but not both. If both are different, then P ⊕ Q will be true otherwise false.
Implication	→	P → Q	If P happens then Q happens.
Double Implication	↔	P ↔ Q	P happens **if and only if** Q happens.

392 ×

Propositional Logic

WORD	SYMBOL	EXAMPLE		TERMINUS TECHNICUS
NOT	¬	not A	¬A	Negation
AND*	∧	A and B	A ∧ B	Conjunction
OR	∨	A or B	A ∨ B	Disjunction
IMPLIES*	→	A implies B	A → B	Implication
IF AND ONLY IF	↔	A if and only if B	A ↔ B	Biconditional

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

State:
Painted(Ceiling) ∧
On(Robot, Ladder)

How would we represent a state where the robot is on the ladder and the ceiling is painted?

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

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

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

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

On(Robot, Floor) ∧
Dry(Ladder) ∧ Dry(Ceiling)
↓ climb-ladder
On(Robot, Ladder) ∧
Dry(Ladder) ∧ Dry(Ceiling)
↓ paint-ceiling
On(Robot, Ladder) ∧
Dry(Ladder) ∧ ¬Dry(Ceiling)
Painted(Ceiling)
↓ descend-ladder
On(Robot, Floor) ∧
Dry(Ladder) ∧ ¬Dry(Ceiling)
Painted(Ceiling)

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

Goal: Painted(Ladder)

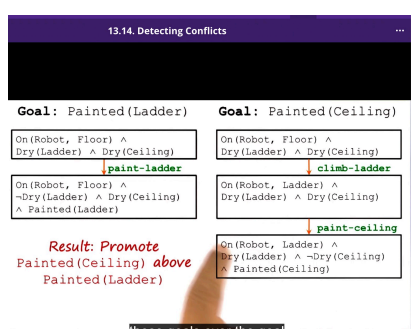
On(Robot, Floor) ∧
Dry(Ladder) ∧ Dry(Ceiling)
↓ paint-ladder
On(Robot, Floor) ∧
¬Dry(Ladder) ∧ Dry(Ceiling)
∧ Painted(Ladder)

Goal: Painted(Ceiling)

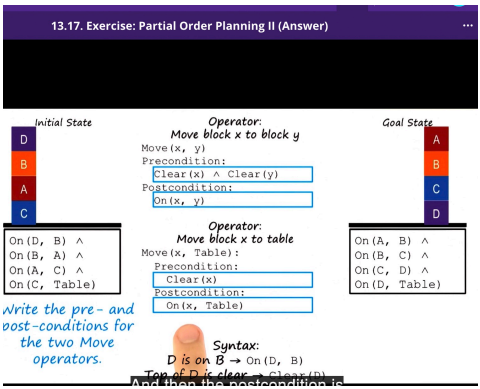
On(Robot, Floor) ∧
Dry(Ladder) ∧ Dry(Ceiling)
↓ climb-ladder
On(Robot, Ladder) ∧
Dry(Ladder) ∧ Dry(Ceiling)
↓ paint-ceiling
On(Robot, Ladder) ∧
Dry(Ladder) ∧ ¬Dry(Ceiling)
∧ Painted(Ceiling)

What operators would be selected and what states would result in order to accomplish the goal Painted(Ceiling)?

But another question



Module 13: Planning

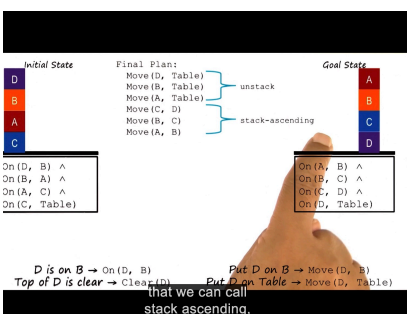


Hierarchical task network planning

- use partial order planning to go from initial to goal state
- can we abstract some of the operations to a higher level?

Hierarchical decomposition

- macro operator



unstack:

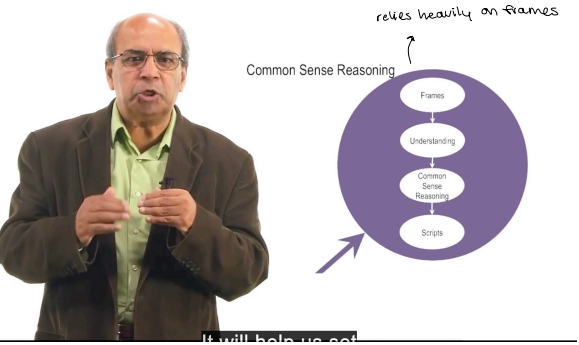
```
Precondition:
  On(w, x) ∧
  On(x, y) ∧
  On(y, z) ∧
  On(z, Table)
Postcondition:
  On(w, Table) ∧
  On(x, Table) ∧
  On(y, Table) ∧
  On(z, Table)
Method:
  Move(w, Table)
  Move(x, Table)
  Move(y, Table)
```

stack-ascending:

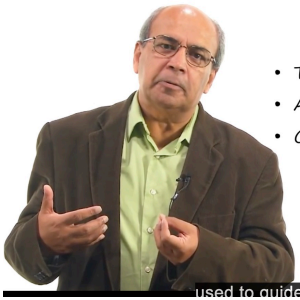
```
Precondition:
  On(a, Table) ∧
  On(b, Table) ∧
  On(c, Table) ∧
  On(d, Table)
Postcondition:
  On(a, b) ∧
  On(b, c) ∧
  On(c, d) ∧
  On(d, Table)
Method:
  Move(c, d)
  Move(b, c)
  Move(a, b)
```

similarly for the stack

Module 14: Understanding



It will help us get



Lesson Preview

- Thematic role systems
- Ambiguity
- Constraints

(a structured type of frame representation)

used to guide our

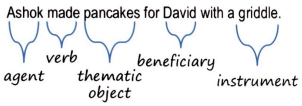
Thematic role systems

- Lexical analysis - which will categorize each of these words into different lexical categories

Ashok made pancakes for David with a griddle
(noun) (verb) (noun) . . .

- Syntactic analysis

(noun phrase) Ashok — (verb phrase) made pancakes for David with a griddle




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

David went to the meeting with Ashok by car.

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

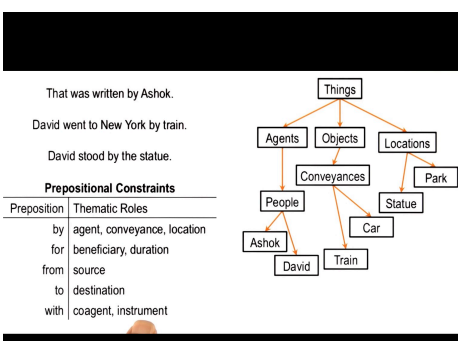
Write the thematic role frame given by the sentence above.

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



That was written by Ashok.

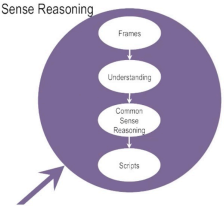
David went to New York by train.

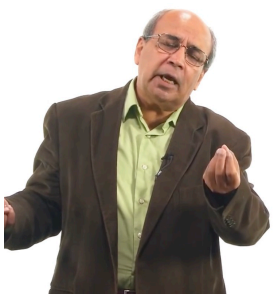
David stood by the statue.

Prepositional Constraints

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

Module 15: Commonsense Reasoning





Lesson Preview

- Primitive actions
- Actions and subactions
- State changes

Primitive Actions

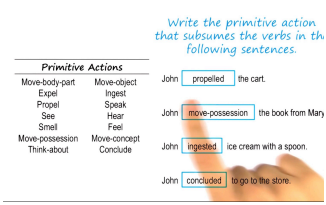
Move-body-part	Move-object
Expel	Ingest
Propel	Speak
See	Hear
Smell	Feel
Move-possession	Move-concept
Think-about	Conclude

Ashok gave idea to Susie about going to a movie → Indecision making

idea in someones mind → mental actions

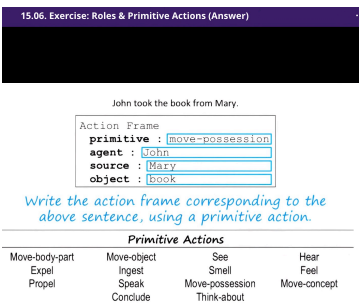
example

- **Ontology - is a specification of a conceptualization of the world**



John pushed the cart.

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



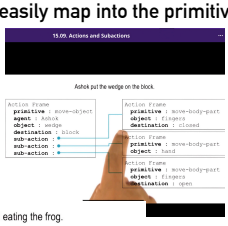
~~John fertilized the field.~~

John put fertilizer on the field.

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

Implied actions

- Sometimes stories only have implied actions in them
- there can be an implied action that is not specified in the structure of the sentence here
- AI agent needs to start thinking in terms of how to transform the sentences bring out implied actions that can more easily map into the primitive actions 15.07 15.08

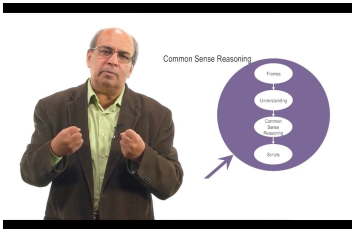


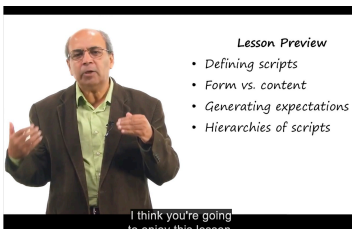
Ashok enjoyed eating the frog.

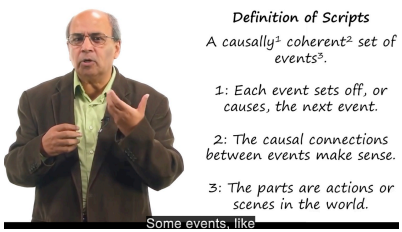
```
Action Frame
  primitive : ingest
  agent : Ashok
  object : frog
  result :
```

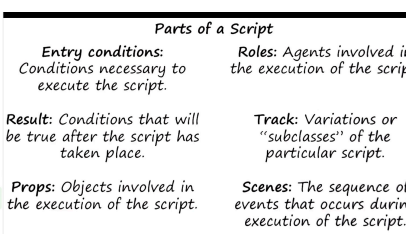
```
State-Change Frame
  object : Ashok's mood
  destination : happy
```

Module 16: Scripts



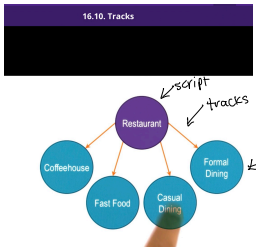


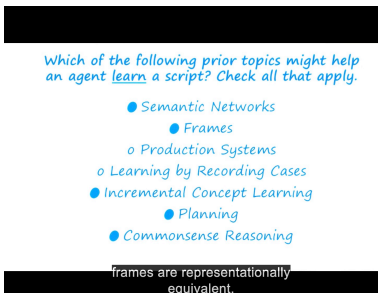




Restaurant Script

```
Script
 script : restaurant
  track : formal dining
  props : tables, menu, check,
          money, F = food, P = place
  roles : S = customer, W = waiter,
          C = cook, M = cashier,
          O = owner
  entry : S is hungry, S has money
 result : S has less money,
          O has more money,
          S is not hungry,
          S is pleased
 scenes :
```

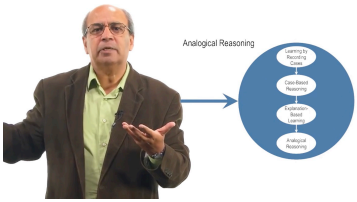


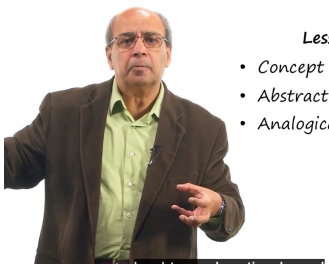


Which of the following prior topics might help an agent use a script? Check all that apply.

- o Generate & Test
- o Means-Ends Analysis
- Problem Reduction
- o Case-Based Reasoning
- Classification
- Logic
- Understanding

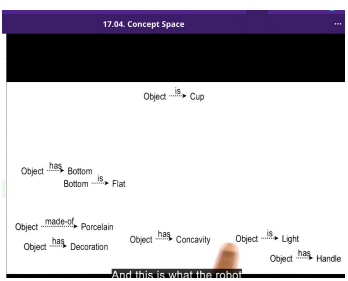
Module 17: Explanation-Based Learning



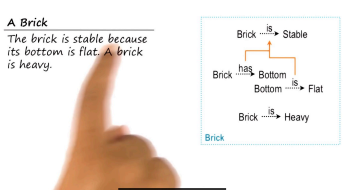


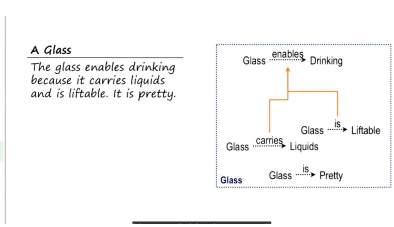
Lesson Preview

- *Concept space*
- *Abstraction*
- *Analogical transfer*

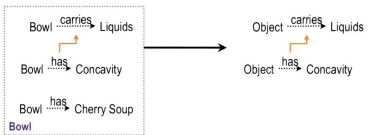


Prior Knowledge

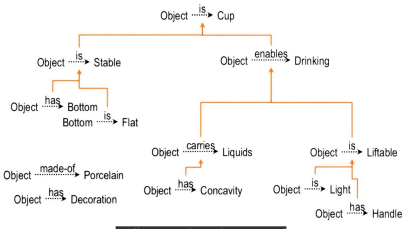


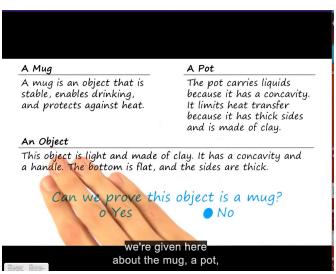


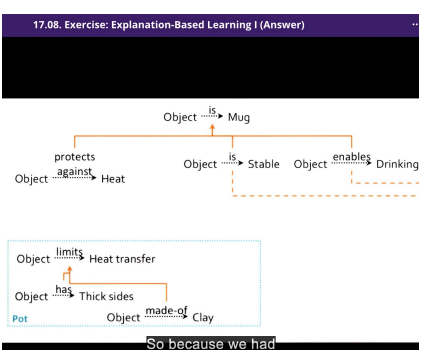
Abstraction



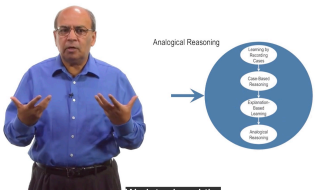
Transfer







Module 18: Analogical Reasoning



Lesson Preview

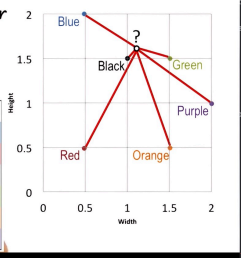
- Similarity and case-based reasoning
- Process of analogical reasoning
- Design by analogy

Finding the Nearest Neighbor

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

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

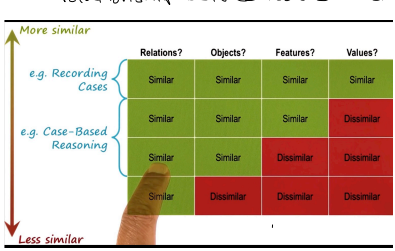
Block	x_c	y_c	x_n	y_n	d
Blue	0.5	2.0	1.1	1.6	0.72
Red	0.5	0.5	1.1	1.6	1.25
Black	**1.0**	**1.5**	**1.1**	**1.6**	**0.14**
Green	1.5	1.5	1.1	1.6	0.41
Orange	1.5	0.5	1.1	1.6	1.17
Purple	2.0	1.0	1.1	1.6	1.08



Cross-domain analogy

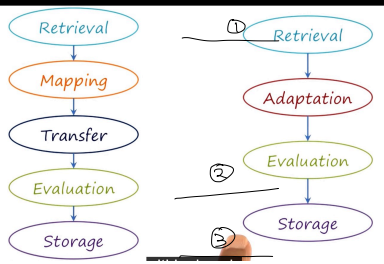
- Target problem: physician and patient
- source case: king and rebel army
- resource broken dawn into smaller resources pointing at the goal
- in cross domain analogies the objects and the features, and the values of the objects can be different
- similarity is based on the relationship
- the relationship is what gets transferred from the source case to the target problem

relationship between source and target



The obvious concepts the source case being the same means that the domains are the same
So case-based reasoning is within domain analogy
An analogical reasoning in general, objects in the target problem and the source case might be different

Process for analogical reasoning



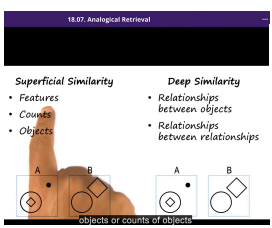
Process for case-based reasoning

18.06

3 commonalities

target problem and the source case need not be from the same domain when they are not from the same domain we can't just take the source case and adapt it. We have to map the target problem to the source — address correspondence problem — what in the target problem corresponds to what in the source case?
(laser beam corresponds to rebel army in the source case)
↓
abstract the relationship
↓
decompose into smaller resources and send to goal

Target problem and source case are from the same domain they have the same relationships and same objects
Cadapt source case to address the target problem)



Analogical retrieval

A woman is climbing a ladder.

Mark whether each situation has deep similarity, superficial similarity, both, or neither with the situation above.

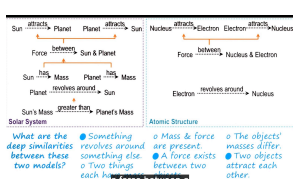
Deep	Superficial	
●	●	A woman climbing a set of stairs.
●	○	An ant walking up the wall.
○	●	A woman painting a ladder.
●	○	A woman climbing the corporate ladder.
○	○	A water bottle sitting on a desk.
●	●	A woman climbing a step ladder.
●	○	A plane taking off into the sky.

Types of Similarity

Semantic *Conceptual similarity between the target problem and the source case.*
between climbing ladder or step ladder

Pragmatic *Similarity of external factors, such as goals.*
physician goal of killing tumor
similar to goal of capturing fortress

Structural *Similarity between representational structures.*

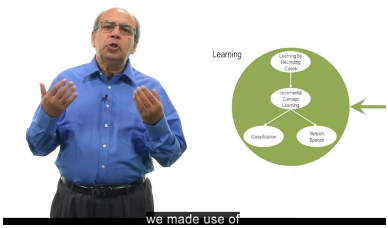


What are the deep similarities between these two models?
○ Something revolves around something else. ○ Two things each h
○ Mass & force are present. ● A force exists between two
○ The objects' masses differ. ● Two objects attract each other.

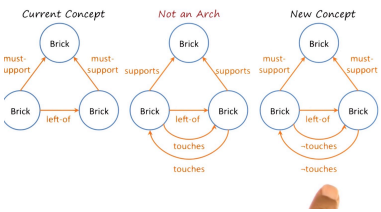
Relationships
higher order relationships — higher precedence
unitary relationship — a patient and king are persons
tertiary relationship — between the good and resource is an obstacle i.e. healthy tissue in a patient w a tumor

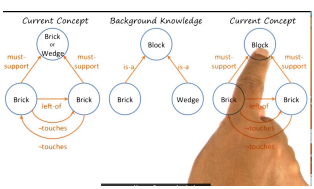
NB!!

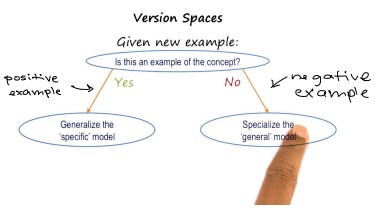
Module 19: Version Spaces



- Negative examples lead to specialization
- positive examples led to generalization







Incremental Concept Learning

(model)
one concept you are moving back and forth

Start with both a specific model and a general model

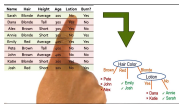
Version Spaces

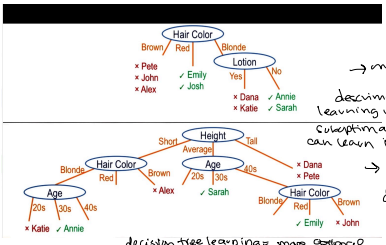
- Convergence is important because without convergence, a learning agent could zig zag forever in a large learning space
- convergence occurs irrespective of the order of the examples
- did not use background knowledge in version spaces as we did with incremental concept learning
- in incremental concept learning we wanted each example to differ from the current concept characterization in exactly are feature so that the learning agent could focus its attention
- in version spaces you notice that each successful example differs from the previous one in many features

Algorithm for Version Spaces

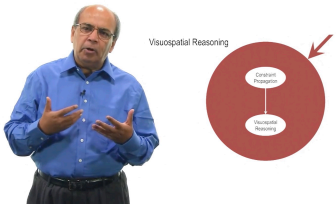
```
For each example:

   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
```

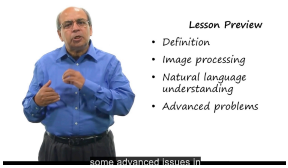
Decision tree learning 19.17





Module 20: Constraint propagation



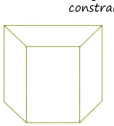


Cross-domain analogy

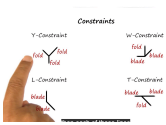
- Target problem: physician and patient
- source case: king and rebel army
- resource broken dawn into smaller resources pointing at the goal
- in cross domain analogies the objects and the features, and the values of the objects can be different
- similarity is based on the relationship
- the relationship is what gets transferred from the source case to the target problem

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

→ does not always succeed in disambiguating between diff assignments and orientations
→ sometimes multiple interpretations can satisfy all the constraints
→ also possible that no assignment of values w/ variables will satisfy all the constraints = difficult to interpret



Colorless green ideas sleep furiously.

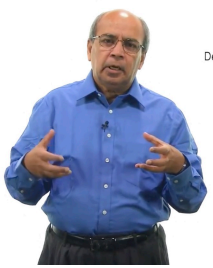


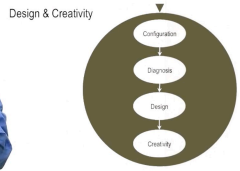
line labeling

knowledge of constraints and constraints propogation

Constraints:
Sentence = Noun Phrase + Verb Phrase
Noun Phrase = [Adjectives] + (Noun or Pronoun)
Verb Phrase = Verb + [Adverb]

Module 21: Configuration

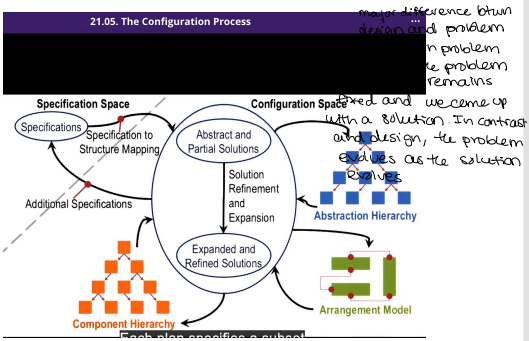


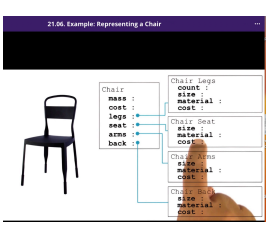


Lesson Preview

- Design & configuration
- Plan refinement
- Connections to earlier topics

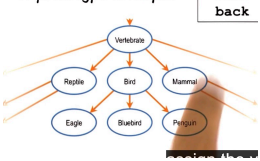
Configuration: A problem-solving activity that assigns values to variables to satisfy constraints.





Classification and configuration are both hierarchical.

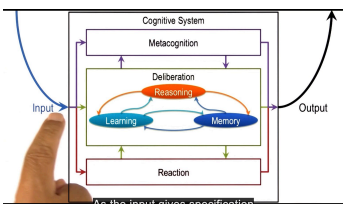
Configuration leverages classification's notion of prototype concepts.



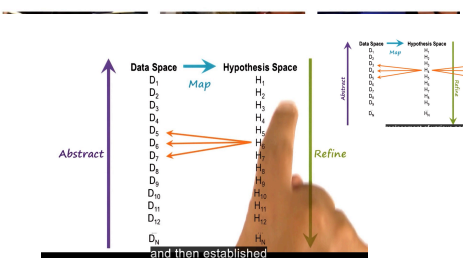
Configuration suggests starting with a prototype concept and assigning values to variables.

Case-based reasoning suggests starting from a specific chair and tweaking it as needed.

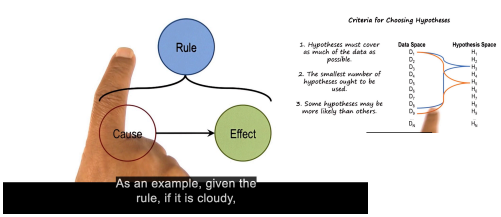
The result of a planning task can lead to a prototype that can subsequently be configured for similar problems with differing constraints.



Diagnosis: To determine what is wrong with a malfunctioning device.



Abduction: Given a rule and an effect, abduce a cause.



- Diagnosis is an instance of abduction
- deduction is truth perserving
- if the rule is true and thpreserving true, we can always guarantee that the effect is true as well
- induction and abduction are not truth preserving - you may know something between cause and effect for some sample but it does not mean the same relationship holds for the entire population

Module 23: Learning by correcting mistakes

Questions for Learning from Mistakes

1. How can the agent isolate the error in its former model?
2. How can the agent explain the problem that led to the error?
3. How can the agent repair the model to prevent the error from recurring?

- identifying error in ones knowledge that led to a failure is called a credit assignment
- incremental learning - one change at a time for the agent to learn from

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)
```

False-Success:

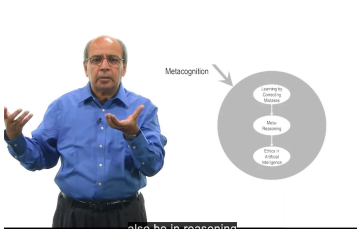
- start off by gathering anything that is ever present in a false success
- then gather everything thats true for every single true success
- remove the things that are true for every success from the things that are ever true for any false success
- then get a list that are true for some false successes

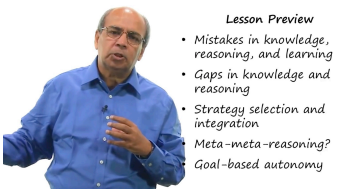
True-success:

- gather everything that is ever true about any true success
- then gather together things that are true for every single false success
- remove everything that every single false example has in common from the things from the things that are true for any true example

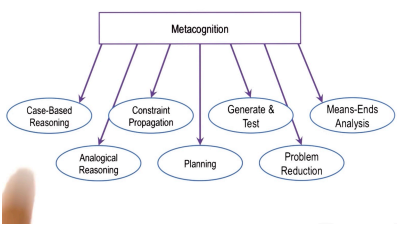
Module 24: Meta Reasoning

- agent is reasoning about its own self/knowledge

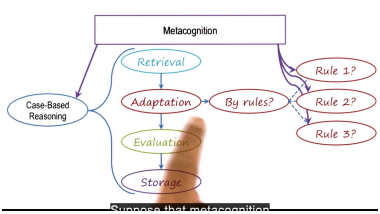




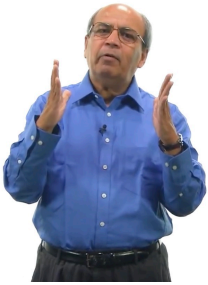
- once an agent detects a gap it can set up a learning goal
- metacognition and deliberation have considerable overlap



- case based reasoning: problem is close to another previous case
- generate and test - if problem is simple, then you can make a generator that will produce good solutions
- means-ends analysis - might be good for single goal, maybe not multiple goals

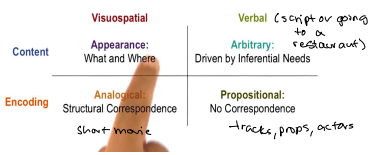


Module 25: Advanced Topics

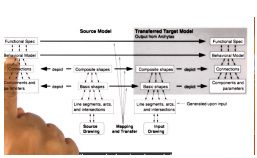


- visual = what part
- spatial = where part

Visuospatial Knowledge:
Knowledge wherein causality is, at most, implicit.



- analogical representation is not the same as analogical reasoning
- some structural correspondence with the external world thats being represented
- modal representations
- close to perceptual modality
- in human cognition, mental imagery appears to use analogical representations
- human cognition is very good at using propositional and analogical representations
- computers are not yet good at using analogical representations - usually only use propositional
- propositional logic is essential for tasks such as knowledge representation, reasoning, and decision-making
- Some examples of analogical representations are maps, pictures, diagrams, linked lists, and flow charts.



Systems Thinking:
Reasoning about systems with numerous components and processes at multiple, potentially invisible, levels of abstraction.

- structure behavior function

Design Thinking:
Reasoning about ill-defined, unconstrained, open problems that are situated in the world.

What is creativity?

A non-obvious, desirable product.

Something is creative if it is...
- Novel
- Valuable
- Unexpected

Some other processes of creativity:
- Emergence
- Re-representation
- Serendipity

Do you agree with David's assessment that none of these results are creative because we can trace through the underlying process that led to them?

- Yes, because in order for a result to be creative, it must be novel, and output of an algorithm cannot be novel.
- Yes, because given a set of input, the output will always be the same; therefore, the product can never be unexpected.
- No, because it defines creativity in terms of the output rather than the process.
- No, because under this definition, humans are only considered creative because we don't know how the brain works yet.

a small closed-word

Module 26: Wrap up

Principle #1: KBAI agents represent and organize knowledge into knowledge structures to guide and support reasoning.

- semantic networks
- frames
- represent organize reason

Principle #2: Learning in KBAI agents is often incremental.

- case-based reasoning
- incremental concept learning
- version spaces
- learning by correcting mistakes

Principle #3: Reasoning in KBAI agents is top-down as well as bottom-up.

- frames = top down reasoning
- scripts = top down reasoning
- constraint propagation =

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

Principle #5: KBAI agents use heuristics to find solutions that are good enough, though not necessarily optimal.

- heuristic search- require link heuristic
- means ends analysis
- generate and test method

Principle #6: KBAI agents make use of recurring patterns in the problems they solve.

- learning by recording cases
- case-based reasoning (within a domain)
- analogical reasoning (spanning domains)

Principle #7: The architecture of KBAI agents enables reasoning, learning, and memory to support and constrain each other.

- chunking in production systems
- explanation based learning
- learning by mistakes

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.