Reasoning in Artificial Intelligence

Reasoning

The reasoning is the psychological procedure of developing logical conclusion and making likelihoods from existing knowledge, facts, and beliefs. Or we can say that, "**Reasoning is a process to generate new knowledge based on previous knowledge.**" It is a common process of thinking reasonably, to find valid inferences.

In artificial intelligence it is used to main machine think like human.

Types of Reasoning

In artificial intelligence, reasoning can be categorized into the following types:

1. **Deductive reasoning**
2. **Inductive reasoning**
3. **Abductive reasoning**
4. **Common Sense Reasoning**
5. **Monotonic Reasoning**
6. **Non-monotonic Reasoning**

1. Deductive reasoning

- In deductive reasoning, the truth of the premises guarantees the truth of the conclusion.

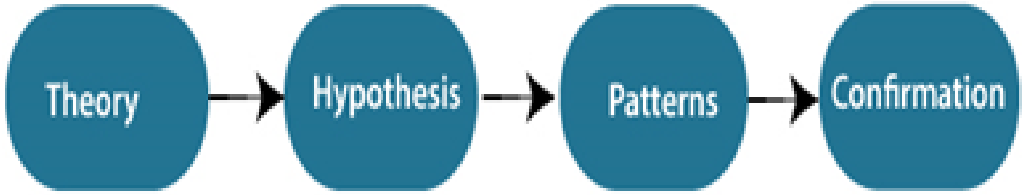
Example:

Premise-1: All the human eats veggies

Premise-2: Suresh is human.

Conclusion: Suresh eats veggies.

- The general process of deductive reasoning is given below:

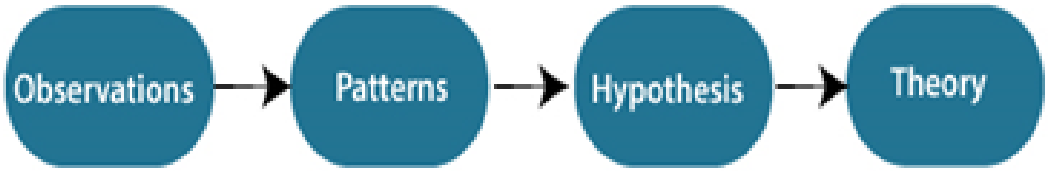


2. Inductive Reasoning:

- Inductive reasoning is a type of propositional logic, which is also known as cause-effect reasoning or bottom-up reasoning.
- In inductive reasoning, premises provide probable supports to the conclusion, so the truth of premises does not guarantee the truth of the conclusion.
- **Example:**

  Premise: All of the pigeons we have seen in the zoo are white.

  Conclusion: Therefore, we can expect all the pigeons to be white.



3. Abductive reasoning

- Abductive reasoning is a form of logical reasoning which starts with single or multiple observations then seeks to find the most likely explanation or conclusion for the observation.
- Abductive reasoning is an extension of deductive reasoning, but in abductive reasoning, the premises do not guarantee the conclusion.
- **Example:**

  Implication: Cricket ground is wet if it is raining

  Axiom: Cricket ground is wet.

  Conclusion: It is raining.

4. Common Sense Reasoning

- Common sense reasoning is an informal form of reasoning, which can be gained through experiences.
- Common Sense reasoning simulates the human ability to make presumptions about events which occurs on every day.
- It relies on good judgment rather than exact logic and operates on **heuristic knowledge** and **heuristic rules.**
- **Example:**
  - **One person can be at one place at a time.**
  - **If I put my hand in a fire, then it will burn.**

5. Monotonic Reasoning

- In monotonic reasoning, once the conclusion is taken, then it will remain the same even if we add some other information to existing information in our knowledge base. In monotonic reasoning, adding knowledge does not decrease the set of prepositions that can be derived.
- To solve monotonic problems, we can derive the valid conclusion from the available facts only, and it will not be affected by new facts.
- Monotonic reasoning is not useful for the real-time systems, as in real time, facts get changed, so we cannot use monotonic reasoning.
- Monotonic reasoning is used in conventional reasoning systems, and a logic-based system is monotonic. *Any theorem proving is an example of monotonic reasoning.*

Example:

Earth revolves around the Sun.

- It is a true fact, and it cannot be changed even if we add another sentence in knowledge base like, "The moon revolves around the earth" Or "Earth is not round," etc.

6. Non-monotonic Reasoning

- In Non-monotonic reasoning, some conclusions may be invalidated if we add some more information to our knowledge base.
- Logic will be said as non-monotonic if some conclusions can be invalidated by adding more knowledge into our knowledge base.
- Non-monotonic reasoning deals with incomplete and uncertain models.
- "Human perceptions for various things in daily life, "is a general example of non-monotonic reasoning.

Example: Let suppose the knowledge base contains the following knowledge:

  - **Birds can fly**
  - **Penguins cannot fly**
  - **Pitty is a bird**

- So from the above sentences, we can conclude that **Pitty can fly**.
- However, if we add one another sentence into knowledge base "**Pitty is a penguin**", which concludes "**Pitty cannot fly**", so it invalidates the above conclusion.

Case Based Reasoning

Faced this situation before?

- Oops the car stopped.
  - What could have gone wrong?
- Aah.. Last time it happened, there was no petrol.
  - Is there petrol?
    - Yes.
  - Oh but wait I remember the tyre was punctured (ban bocor)
- This is the normal thought process of a human when faced with a problem which is similar to a problem he/she had faced before.

So what?

- Reuse the solution experience when faced with a similar problem.
- This is Case Based Reasoning (CBR)!
  - memory-based problem-solving
  - re-using past experiences
- Experts often find it easier to relate stories about past cases than to formulate rules

What's CBR?

- To solve a new problem by remembering a previous similar situation and by reusing information and knowledge of that situation
- Ex: Medicine
  - doctor remembers previous patients especially for rare combinations of symptoms
- Ex: Law
  - English/US law depends on precedence
  - case histories are consulted
- Ex: Management
  - decisions are often based on past rulings
- Ex: Financial
  - performance is predicted by past results

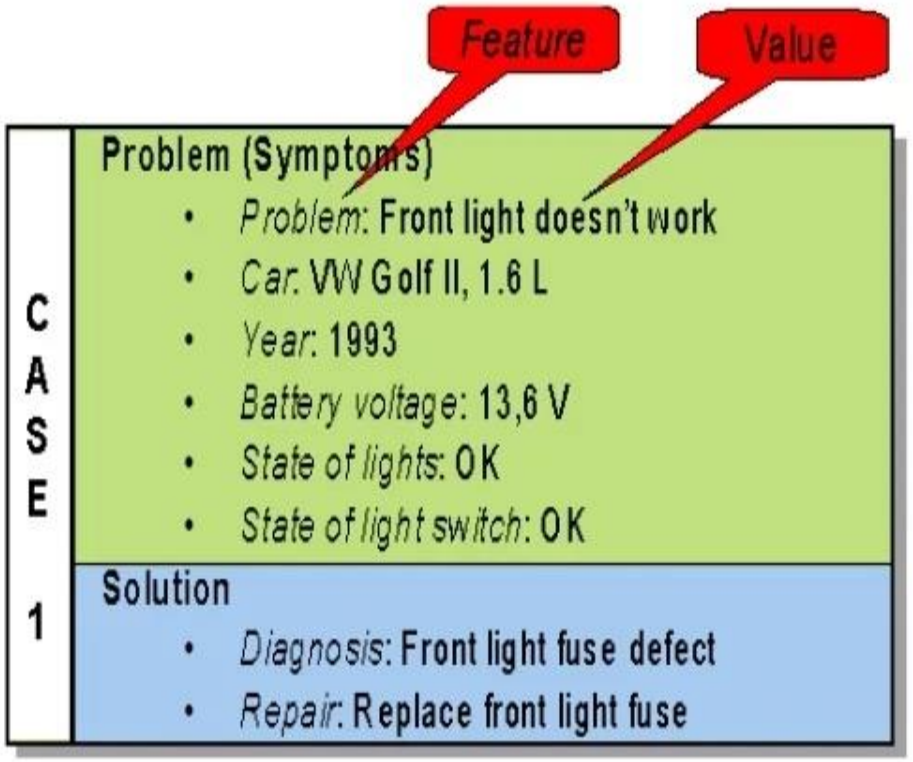
CBR System Components

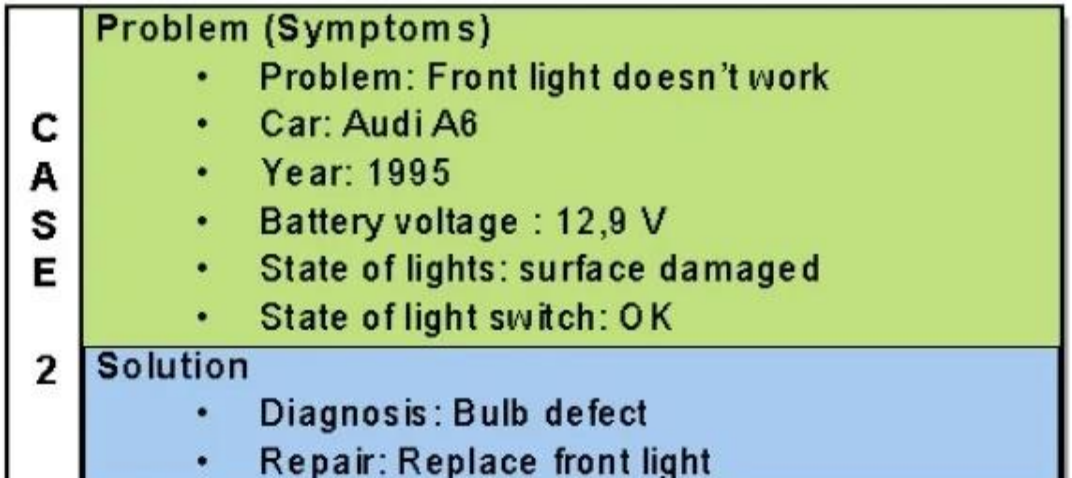
- Case-base
  - database of previous cases (experience)
- Retrieval of relevant cases
  - index for cases in library
  - matching most similar case(s)
  - retrieving the solution(s) from these case(s)
- Adaptation of solution
  - alter the retrieved solution(s) to reflect differences between new case and retrieved case(s)

Two big tasks of CBR

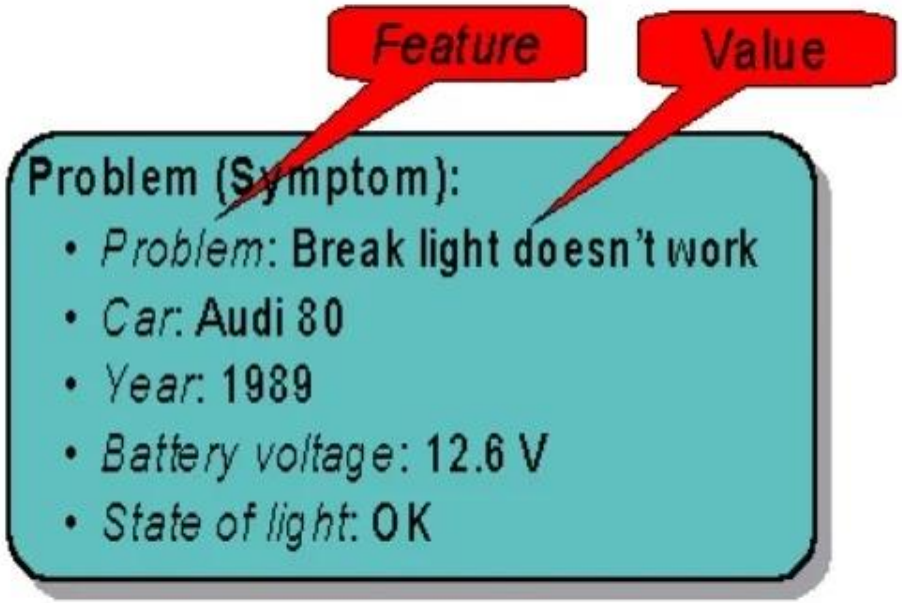
- Classification tasks (good for CBR)
  - Diagnosis - what type of fault is this?
  - Prediction / estimation - what happened when we saw this pattern before?
- Synthesis tasks (harder for CBR)
  - Engineering Design
  - Planning
  - Scheduling

Technical Diagnosis of Car Faults



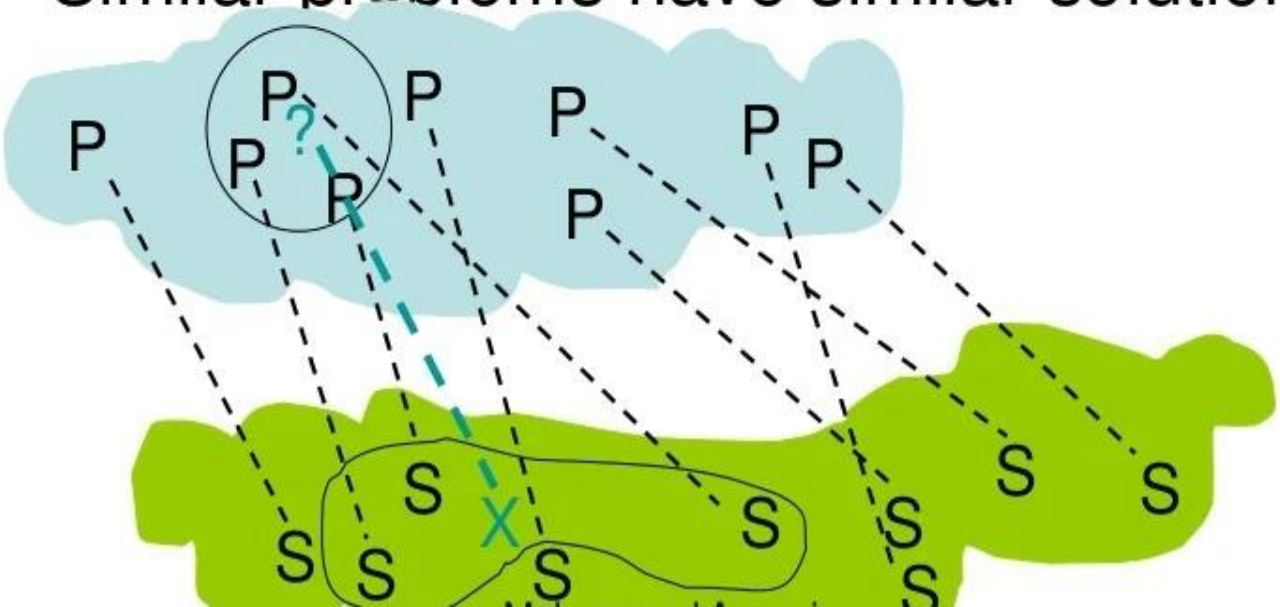


Problem to be solved



How CBR solves problems

- New problem can be solved by
  - retrieving similar problems
  - adapting retrieved solutions
- Similar problems have similar solutions

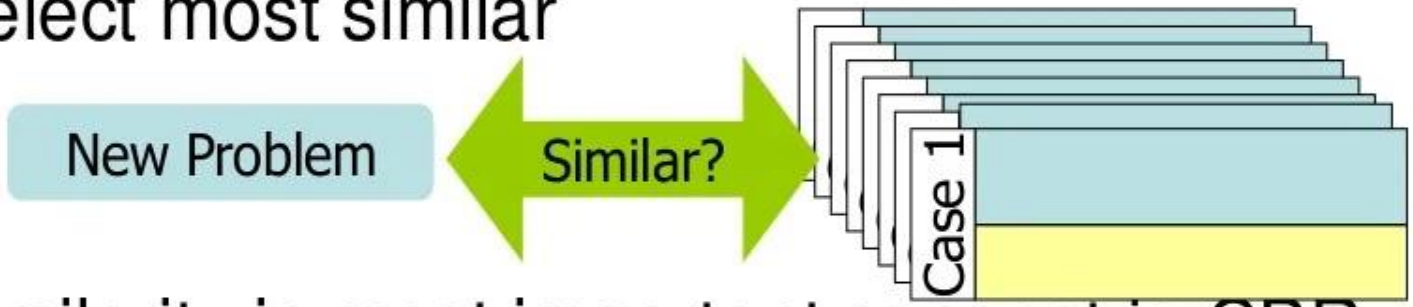


New Car Diagnosis Problem

- A new problem is a case without a solution part
- Not all problem features must be known
  - same for cases

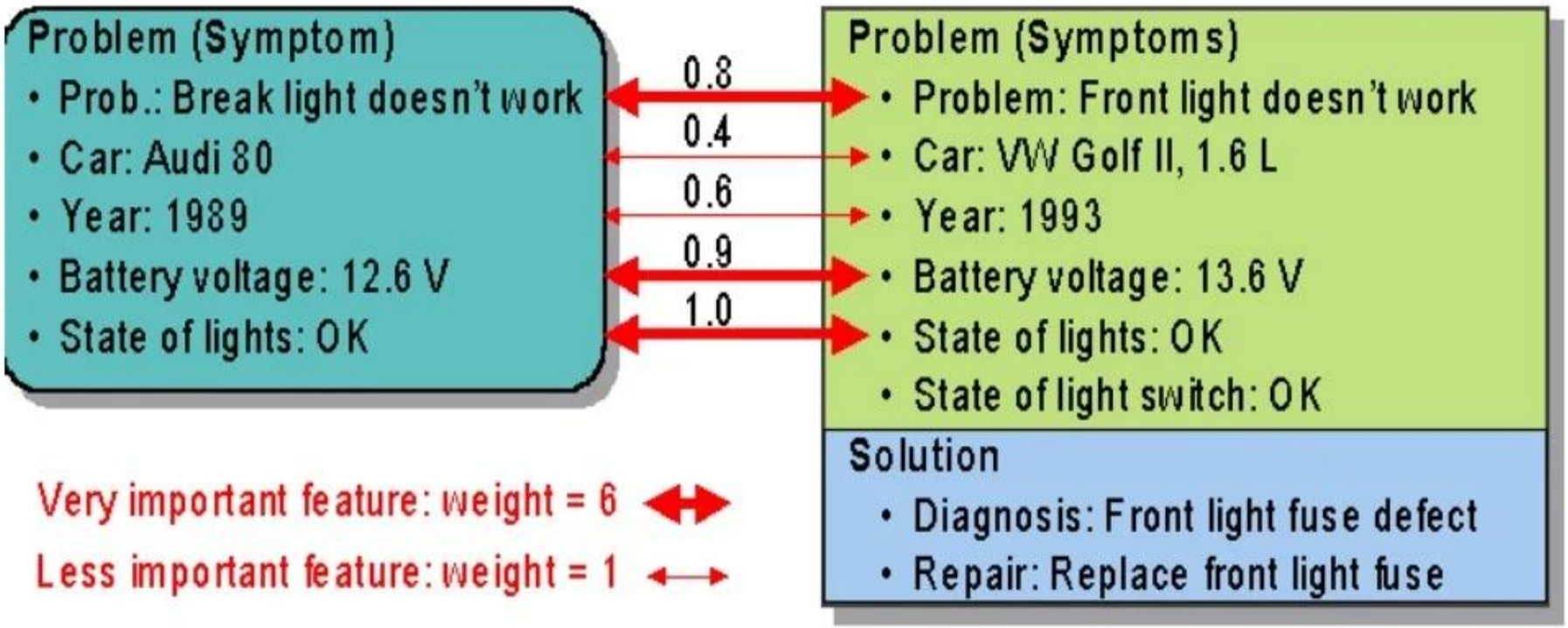
Retrieving A Car Diagnosis Case

- Compare new problem to each case
- Select most similar



- Similarity is most important concept in CBR
  - When are two cases similar?
  - How are cases ranked according to similarity?
- Similarity of cases
  - Similarity for each feature
    - Depends on feature values

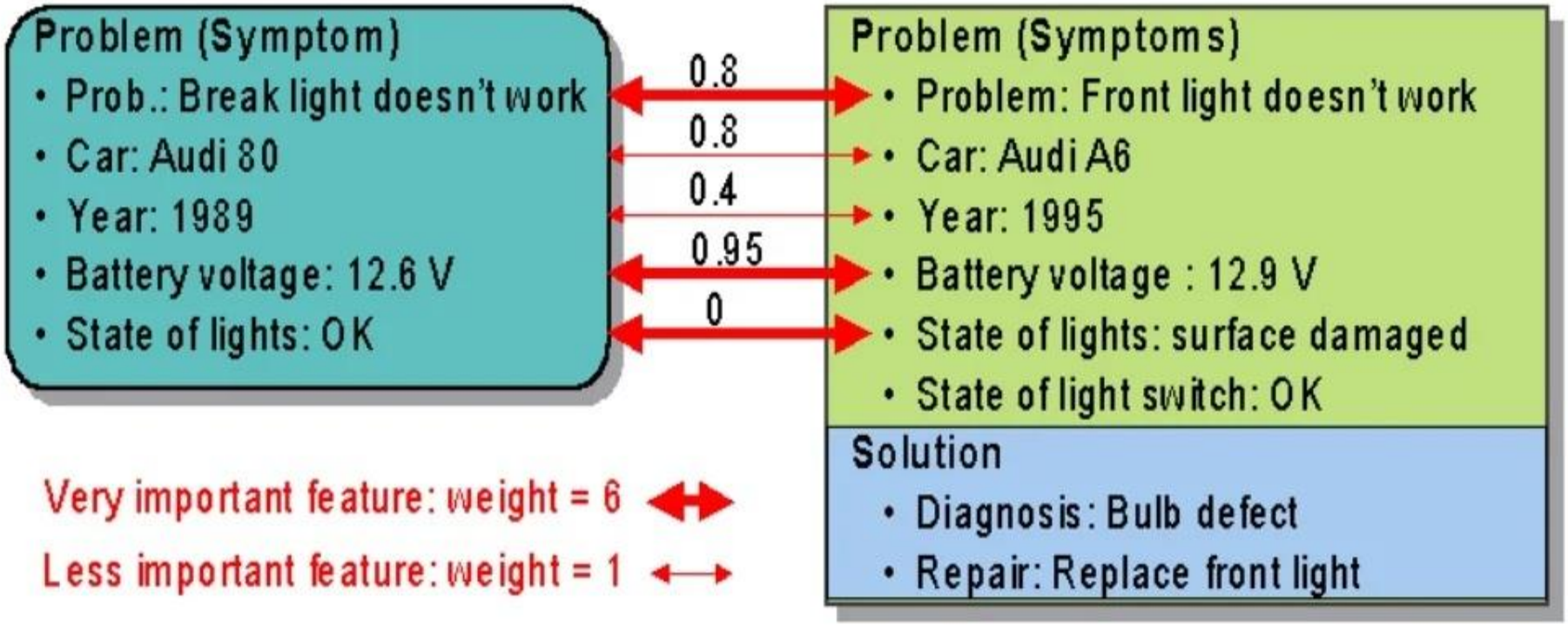
Similarity Computation for case 1



Similarity Computation by Weighted Average

similarity(new,case 1) = 1/20 * [6*0.8 + 1*0.4 + 1*0.6 + 6*0.9 + 6* 1.0] = **0.86**

Similarity Computation for case 2



Similarity Computation by Weighted Average

similarity(new,case 2) = 1/20 * [6*0.8 + 1*0.8 + 1*0.4 + 6*0.95 + 6*0] = **0.585**

Similarity Measurement

- Purpose: To select the most relevant case
- Basic Assumption: Similar problems have similar solutions
- Similarity value between 0 and 1 are assigned for feature value pairs
- E.g.: Feature: Problem

Front Light does not work

.8

Break Light does not work

Front Light does not work

.4

Engine doesn't start

Similarity Measurement

- Feature: Battery Voltage

$$12.6 \overset{.9}{=} 13.6 \qquad 12.6 \overset{.1}{=} 6.7$$

- Different features have different importance
- Two kinds of Similarity Measures
  - Local Similarity – similarity on feature level
  - Global Similarity - similarity on case or object level

Reuse Solution from Case 1

New Problem

- Symptom: brakelight does not work
- Car: Ford Fiesta
- Year: 1997
- Battery: 9.2v
- Headlights: undamaged
- HeadlightSwitch: ?

Problem Case 1

- Symptom: headlight does not work
- …

Solution

- Diagnosis: headlight fuse blown
- Repair: replace headlight fuse

– Solution to New Problem
  - Diagnosis: headlight fuse blown
  - Repair: replace headlight fuse

– After Adaptation
  - Diagnosis: brakelight fuse blown
  - Repair: replace brakelight fuse

Pros & Cons of CBR

- Advantages
  - solutions are quickly proposed
    - derivation from scratch is avoided
  - domains do not need to be completely understood
  - cases useful for open-ended/ill-defined concepts
  - highlights important features
- Disadvantages
  - old cases may be poor
  - library may be biased
  - most appropriate cases may not be retrieved
  - retrieval/adaptation knowledge still needed

Analogical reasoning

Analogy

- At its most fundamental level, an analogy is a comparative relationship between two (or more) different things or ideas.
- Analogies draw attention to how two unlike things are similar. As a cognitive tool, they allow us to infer the properties or predict the behavior of an unknown entity based on its similarities to a known entity. Analogical reasoning is one of the most important cognitive tools we use to structure our understanding of and beliefs about the world. Below are some of the crucial contexts in which analogies function:

- Cognitive Development: Analogies play an important role in human cognitive and linguistic development; children compare their knowledge about past experiences to make predictions about present or future experiences.

- Models and Maps: Mental and physical models are created for the express purpose of illustrating analogies. Maps and models depend on their similarities to and differences from whatever it is they are supposed to represent. A map of Europe would not be practical if it was the size of the continent itself; instead, a good map has consistent and accurate spatial relationships between points of interest.

- Scientific Method: Scientists use inductive reasoning, which involves recognizing analogies between things that are known and things that are hypothesized but not yet known. If the unknown thing is similar in important respects to a known thing, they make inferences about the unknown thing based on its similarities to the known thing. The Italian scientist Galileo used simple Archimedean machines like pendulums and inclined planes as analogies for more complex phenomena like freefall. While developing his laws of planetary motion, Johannes Kepler was inspired by analogies from musical theory, like harmony, octave, and pitch.

- Literary Devices: Poets and storytellers use specific forms of analogy to create vivid imagery. A metaphor is an analogy that states that one thing is another thing to make some feature explicit. A simile is an analogy that compares two things by stating that they are like each other.
- "The clock is a merciless master" is an example of a metaphor.
- "The sickness came like a thief" in the night is an example of a simile.

- Legal Precedent: When courts rule on a case, they set precedent. The ever-growing body of legal precedent is used as a reference when courts are presented with legal disputes. Present cases are compared to past cases, and if a case is analogous to precedent, courts tend to rule as they ruled in the past.

Analogical Reasoning

- In informal terms, to reason by analogy is merely to compare two unlike things and to make inferences based on the resulting analysis. Logicians and philosophers, however, have a stricter definition of analogical reasoning: reasoning by analogy means to use an analogical argument to form a conclusion. An analogical argument is a formal argumentative form, the structure of which is described in the following section.
- The primary difference between deductive arguments and analogical arguments is that if a deductive argument is valid and sound, the truth of the conclusion is guaranteed. Analogical arguments are called ampliative, meaning that the conclusion may be false even if the argument is valid and sound.
- Although analogical arguments are not truth-guaranteeing, they allow the possibility of inference, which leads to new knowledge. For this reason, analogical arguments are extremely important in domains like science.

Structure of Analogical Reasoning

Suppose that a child graduates from elementary school and moves on to high school. Although she doesn't know for certain what it will be like, she can reason analogically to infer certain likelihoods about her upcoming experience. Formally, her analogical argument might look as follows:

P. Both schools have teachers, a gymnasium, and classrooms.
P. The elementary school has a football field.
C. Therefore, the high school also has a football field.

Although the student can know with certainty that both premises are true, analogical arguments do not guarantee the truth of their conclusions. A slightly modified version of the example demonstrates how this is the case:

P. Both schools have teachers, a gymnasium, and classrooms.
P. The elementary school has cubbies for coats and lunchboxes.
C. Therefore, the high school has cubbies for coats and lunchboxes.

The above is a valid analogical argument, and its premises are true. But chances are, the high school will have lockers instead of cubbies.

The relata of an analogical argument are called the source domain and the target domain. In the example, the elementary school is the source domain and high school is the target domain; known properties of the source domain are inferred about the target domain based on known similarities.

Source Domain	Target Domain
Elementary School	High School
Known	Inferred

Type of Reasoning

- Content free reasoning:
  - Deductive reasoning
- Reasoning by similarity
  - Inductive reasoning
  - Analogical reasoning:
    - analogical transfer: solving problem in one doma based on solution in another domain
    - Analogical inference: generalizing properties/relations from one domain to another

Analogical transfer

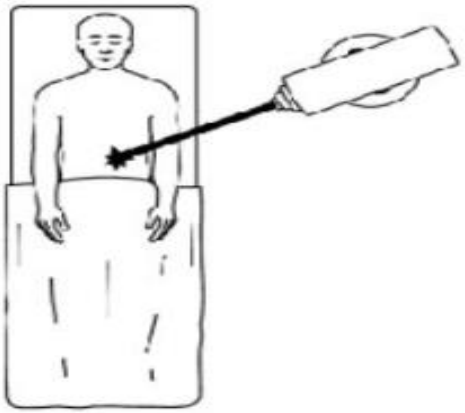
Reason from base problem (previously solved) to target problem

- **Recognition**: identify a potential analog
- **Abstraction**: abstract general principle from base problem
- **Mapping:** apply principle to target

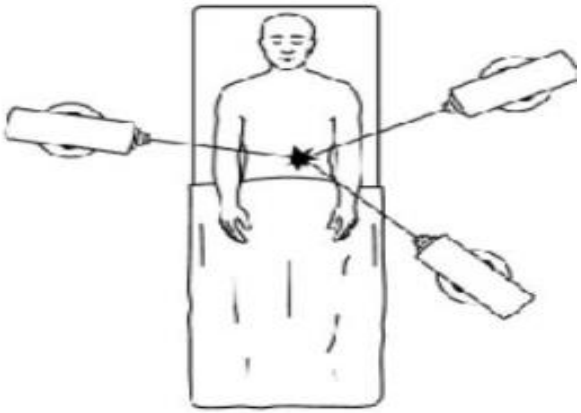
Radiation Problem

- Problem
  - Patient has an abdominal tumor (in center of body)
  - Radiation will kill tumor
  - But rays strong enough to kill tumor will also destroy healthy tissue that rays pass through on way to tumor
- Solution?

Solution



INCORRECT



CORRECT

Gick & Holyoak (1980)

Only 20% solved it

Participants were more likely to solve the problem if they were given an analogy.

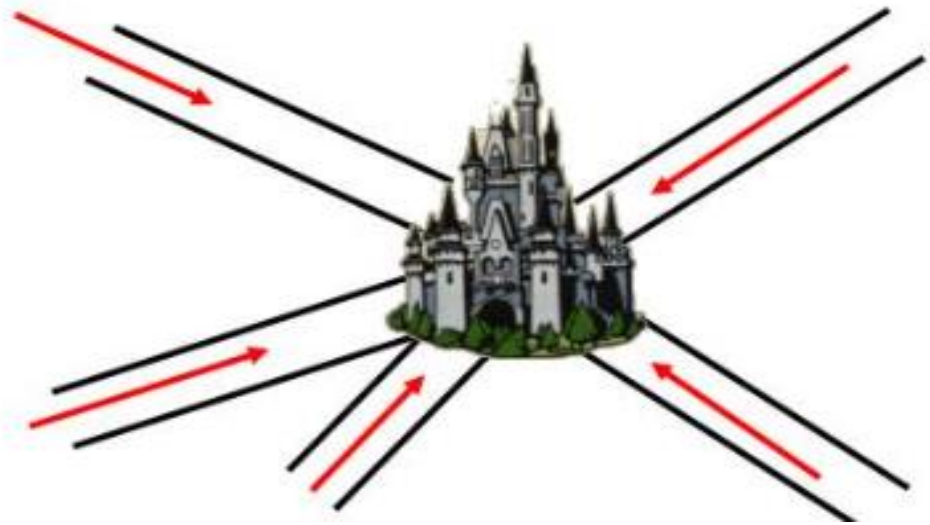
Fortress Problem

- Army general wants to attack a fortress in center of forest.
- Many roads lead to fortress like spokes of a wagon wheel.
- All the roads have land mines that are triggered by heavy traffic.
- If entire army travels down a road, land mines will go off.

Solution?

Solution

The general sends small groups of soldiers down each of the roads, co-ordinating their movements so that they all arrive at the fortress at the same time.



Structural Similarities

Fortress Problem		Radiation Problem
Fortress	←→	Tumor
Mined Roads	←→	Surrounding tissue
Attacking troops	←→	Rays varying in intensity
Small groups of troops Converging on fortress	←→	Weak rays focused on tumor

Recognition Process

- Radiation problem hard problem:
  - Preceded by Fortress problem, 20% solved it
  - If given hint that problems are related, 92% solved it
- Subjects do not produce corresponding solutions with any great frequency unless given a hint
- These results demonstrate the difficulty of recognizing the relevance of one problem and its solution to another

Transfer aspects of fortress problem to radiation problem

- Fortress problem was given with three different solutions:
  - **Dispersion**: the general's army approaching the fort from several roads at once.
  - **Tunnel**: a general digging a tunnel to the fort
  - **Open-route**: a general attacking the fort via a supply route

Problem Solutions

Solutions to the Radiation problem fell into three main categories:

The DISPERSION solution, in which small doses of radiation are aimed at the tumor from different directions so that the radiation accumulates and destroys the tumorous tissue;

The TUNNEL solution, in which an incision is made which leaves the tumor in place but which allows direct delivery of radiation.

The OPEN-ROUTE solution, in which radiation is delivered to the tumor through the esophagus (tube leading from the throat to the stomach);

Distribution (%) of Solutions

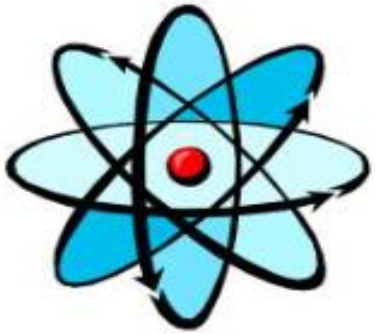
Base Problem Solution	Target Problem Solution		
	Dispersion	Open Route	Operation
Dispersion	**100**	10	30
Open Route	10	**70**	50
Tunnel	20	30	**80**
None	0	20	50

What do the results show?

Abstraction Process

- Does performance improve if principle of base problem is stated? No
- Showing diagrams helps? No
- Showing multiple problems with related solutions helps? Yes

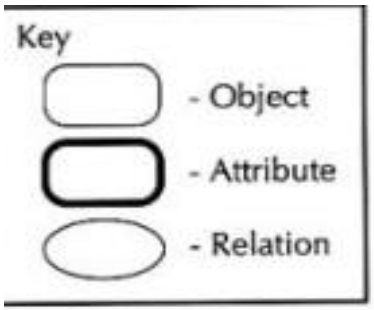
Analogical Inference

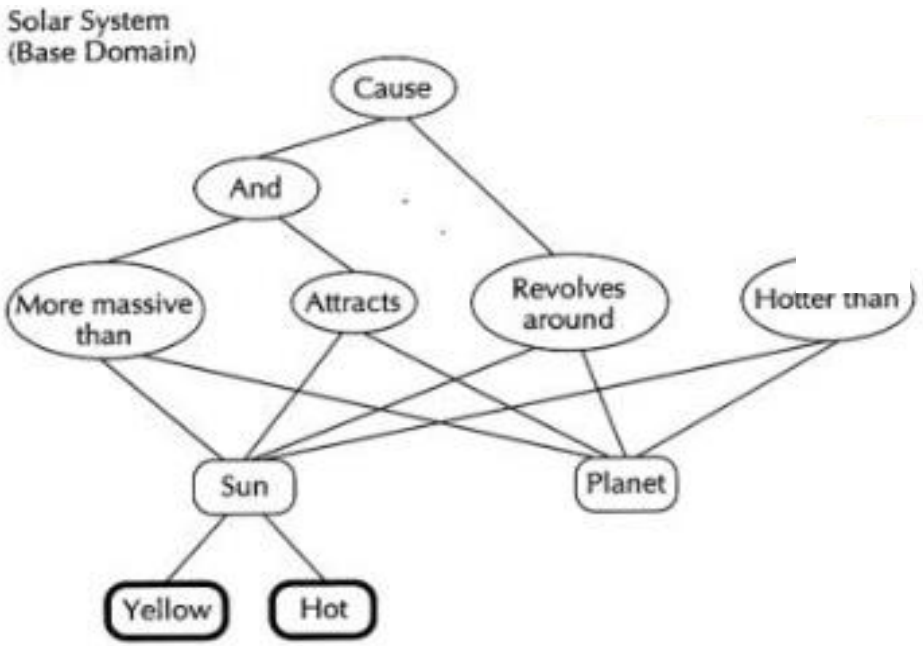


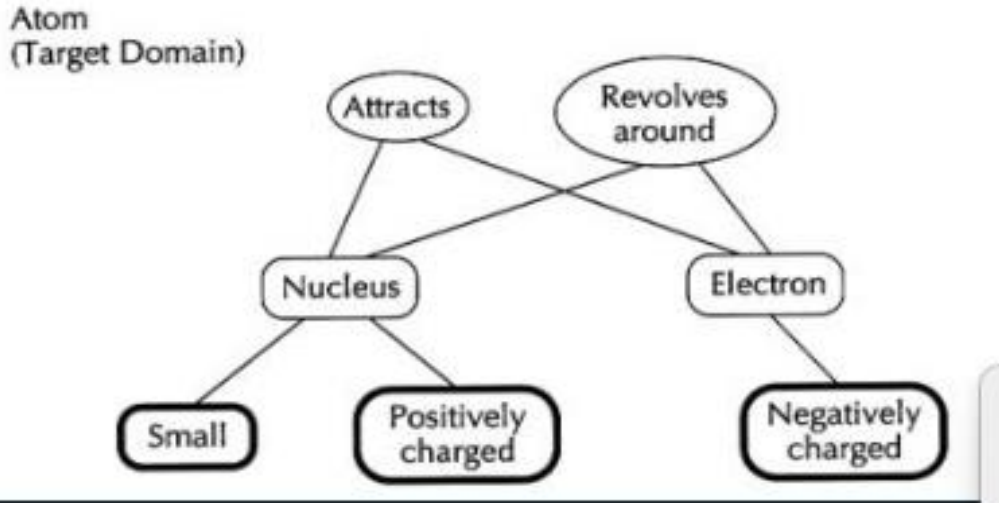
"An atom is like the solar system"

TARGET BASE

Knowledge about the base domain can be used to reason about the target domain. Structure mapping theory is a theory for how this could work.


Key
- Object
- Attribute
- Relation


Solar System
(Base Domain)
Cause
And
More massive than
Attracts
Revolves around
Hotter than
Sun
Planet
Yellow
Hot

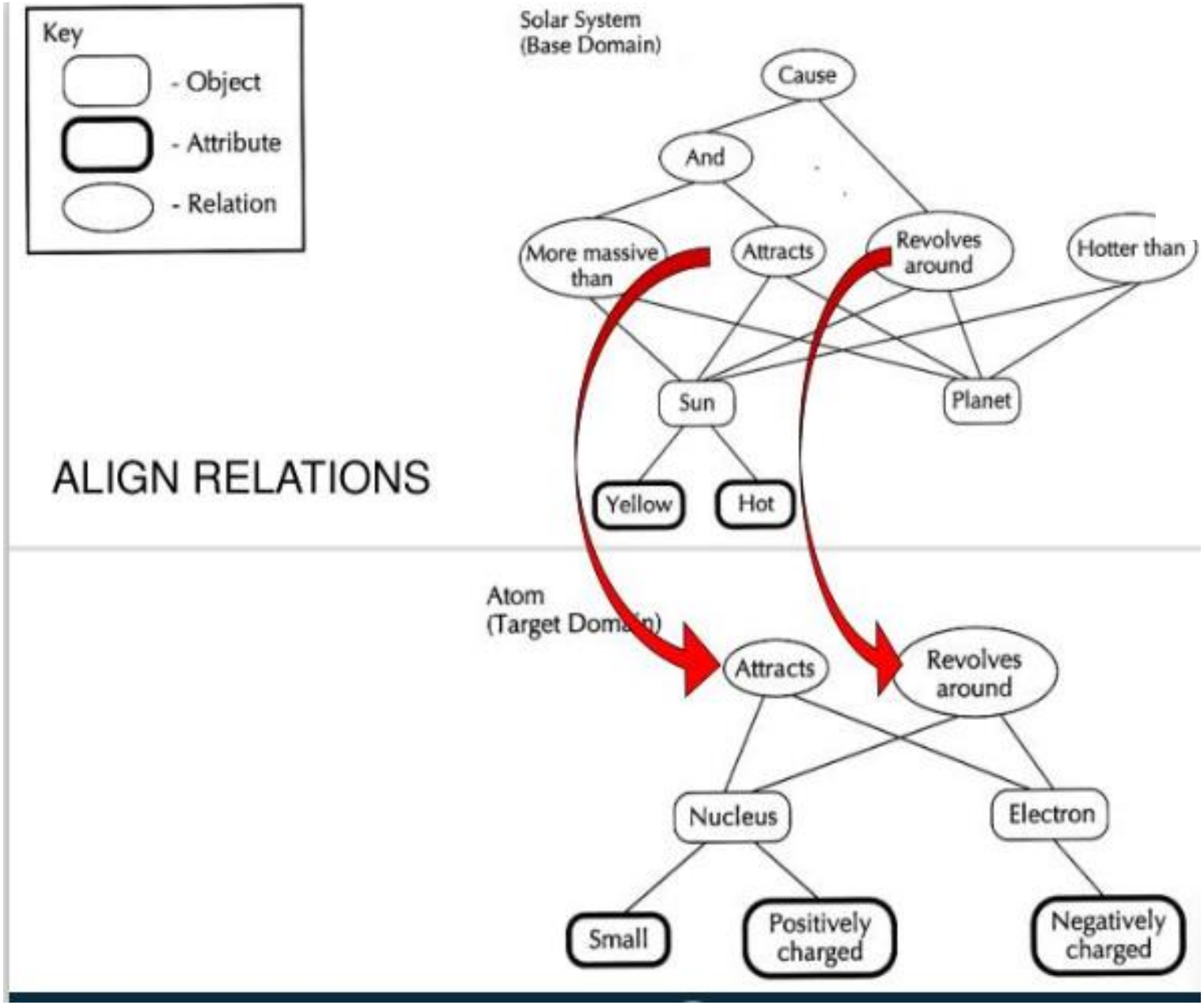

Atom
(Target Domain)
Attracts
Revolves around
Nucleus
Electron
Small
Positively charged
Negatively charged

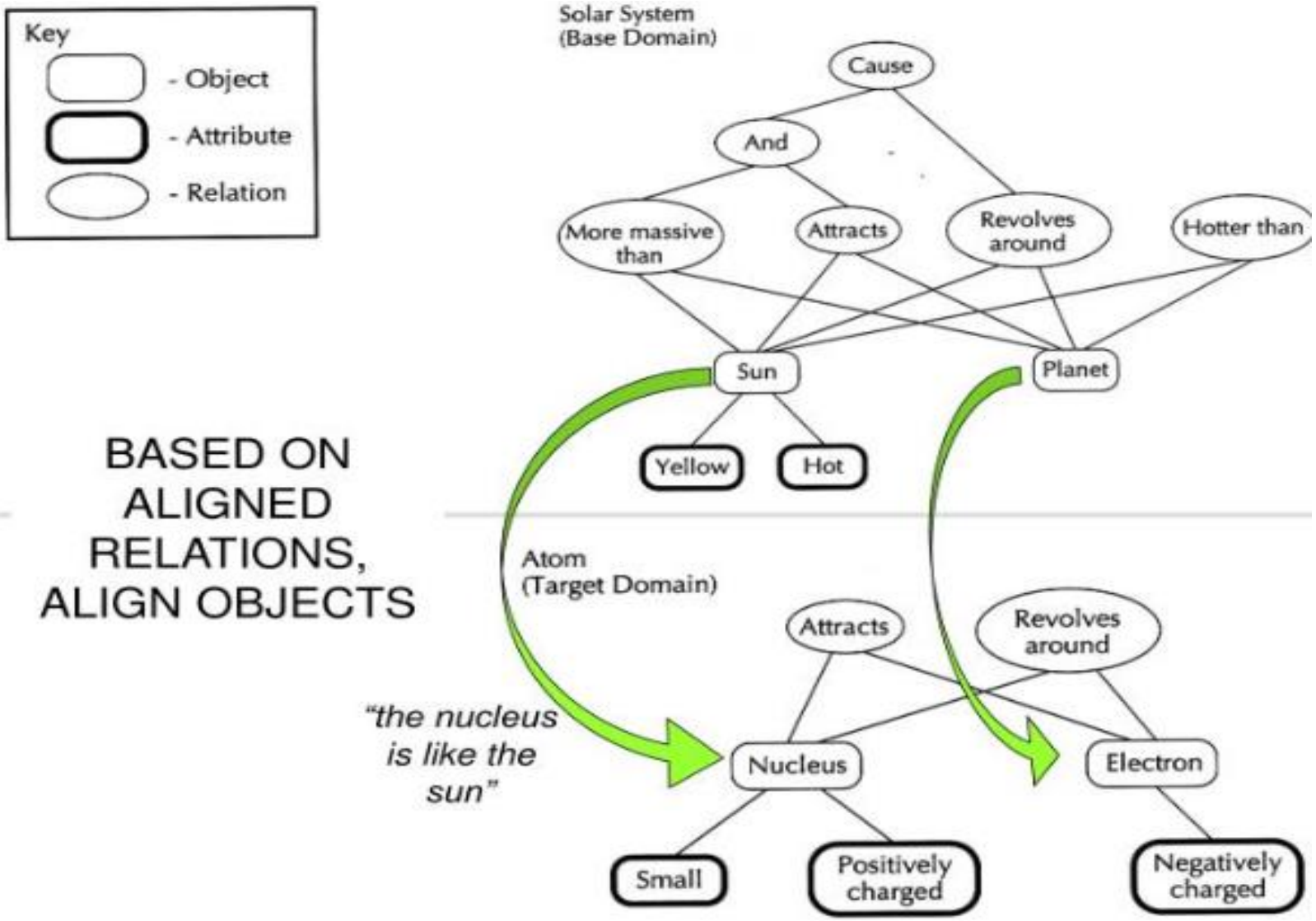
Structure Mapping Theory

(Dedre Gentner, '83, '89)

Theory on how to interpret an analogy

1) set up correspondences
2) focus on matches of relations, not attributes
3) focus on higher-order relations
4) extend knowledge about target by mapping relations from source to target


Key
- Object
- Attribute
- Relation
Solar System
(Base Domain)
Cause
And
More massive than
Attracts
Revolves around
Hotter than
Sun
Planet
Yellow
Hot
ALIGN RELATIONS
Atom
(Target Domain)
Attracts
Revolves around
Nucleus
Electron
Small
Positively charged
Negatively charged


Key
- Object
- Attribute
- Relation
Solar System
(Base Domain)
Cause
And
More massive than
Attracts
Revolves around
Hotter than
Sun
Planet
Yellow
Hot
BASED ON ALIGNED RELATIONS, ALIGN OBJECTS
Atom
(Target Domain)
Attracts
Revolves around
"the nucleus is like the sun"
Nucleus
Electron
Small
Positively charged
Negatively charged

Constraint Satisfaction

Overview

- Constraint satisfaction problems (CSPs) need solutions that satisfy all the associated constraints. Look into the definition and examples of constraint satisfaction problems and understand the process of converting problems to CSPs, using examples.
- Consider a Sudoku game with some numbers filled initially in some squares. You are expected to fill the empty squares with numbers ranging from 1 to 9 in such a way that no row, column or a block has a number repeating itself. This is a very basic constraint satisfaction problem. You are supposed to solve a problem keeping in mind some constraints. The remaining squares that are to be filled are known as variables, and the range of numbers (1-9) that can fill them is known as a domain. Variables take on values from the domain. The conditions governing how a variable will choose its domain are known as constraints.

Overview

A constraint satisfaction problem (CSP) is a problem that requires its solution within some limitations or conditions also known as constraints. It consists of the following:

•A finite set of variables which stores the solution (V = {V1, V2, V3,....., Vn})
•A set of discrete values known as domain from which the solution is picked (D = {D1, D2, D3,.....,Dn})
•A finite set of constraints (C = {C1, C2, C3,....., Cn})

Please note, that the elements in the domain can be both continuous and discrete but in AI, we generally only deal with discrete values.

Also, note that all these sets should be finite except for the domain set. Each variable in the variable set can have different domains. For example, consider the Sudoku problem again. Suppose that a row, column and block already have 3, 5 and 7 filled in. Then the domain for all the variables in that row, column and block will be {1, 2, 4, 6, 8, 9}.

Popular Problems with CSP

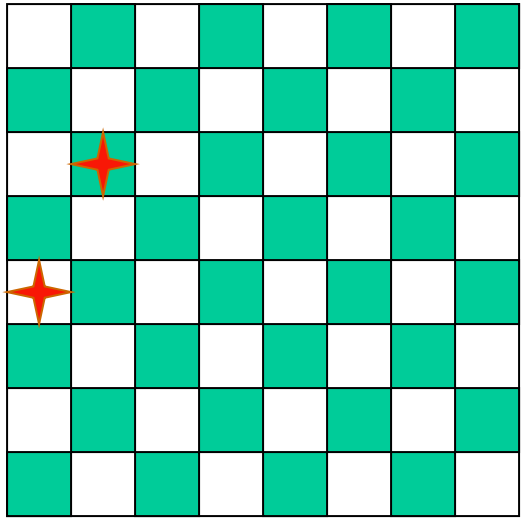
The following problems are some of the popular problems that can be solved using CSP:

1.CryptArithmetic (Coding alphabets to numbers.)
2.n-Queen (In an n-queen problem, n queens should be placed in an nXn matrix such that no queen shares the same row, column or diagonal.)
3.Map Coloring (coloring different regions of map, ensuring no adjacent regions have the same color)
4.Crossword (everyday puzzles appearing in newspapers)
5.Sudoku (a number grid)
6.Latin Square Problem

- Algorithms for CSPs
  - Backtracking (systematic search)
  - Constraint propagation (k-consistency)
  - Variable and value ordering heuristics
  - Backjumping and dependency-directed backtracking

Motivating example: 8 Queens

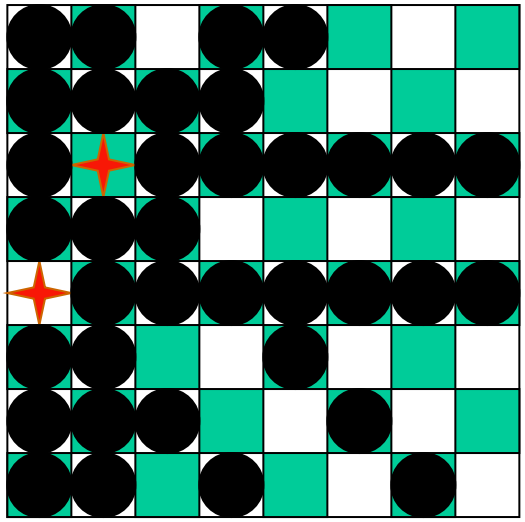
Place 8 queens on a chess board such
That none is attacking another.



Generate-and-test, with no redundancies → "only" 8^8 combinations

*8**8 is 16,777,216*

Motivating example: 8-Queens



After placing these two queens, it's trivial to make the squares we can no longer use

What more do we need for 8 queens?

- Not just a *successor function* and *goal test*
- But also
  - a means to *propagate constraints* imposed by one queen on the others
  - an *early failure test*

→ Explicit representation of constraints and constraint manipulation algorithms

Informal definition of CSP

- CSP = Constraint Satisfaction Problem, given

  (1) finite set of variables

  (2) each with domain of possible values (often finite)

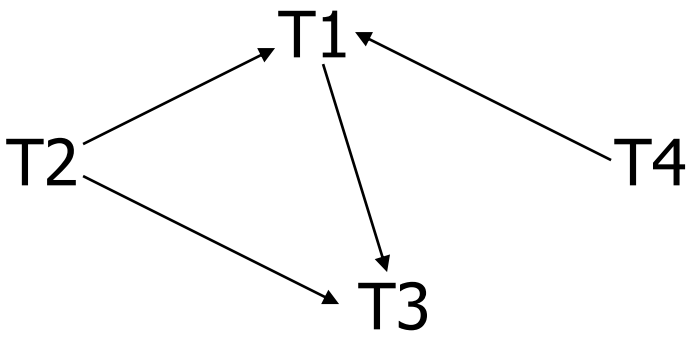
  (3) set of constraints limiting values variables can take on

- S**olution** is an assignment of a value to each variable such that the constraints are all satisfied
- Tasks might be to decide if a solution exists, to find a solution, to find all solutions, or to find “best solution” according to some metric (objective function).

Example: 8-Queens Problem

- Eight variables Xi, i = 1..8 where Xi is the row number of queen in column i
- Domain for each variable {1,2,…,8}
- Constraints are of the forms:
  - Not on same row:
    $X_i = k \rightarrow X_j \neq k$ for $j = 1..8, j \neq i$
  - Not on same diagonal
    $X_i = k_i, X_j = k_j \rightarrow |i-j| \neq |k_i - k_j|$ for $j = 1..8, j \neq i$

Example: Task Scheduling

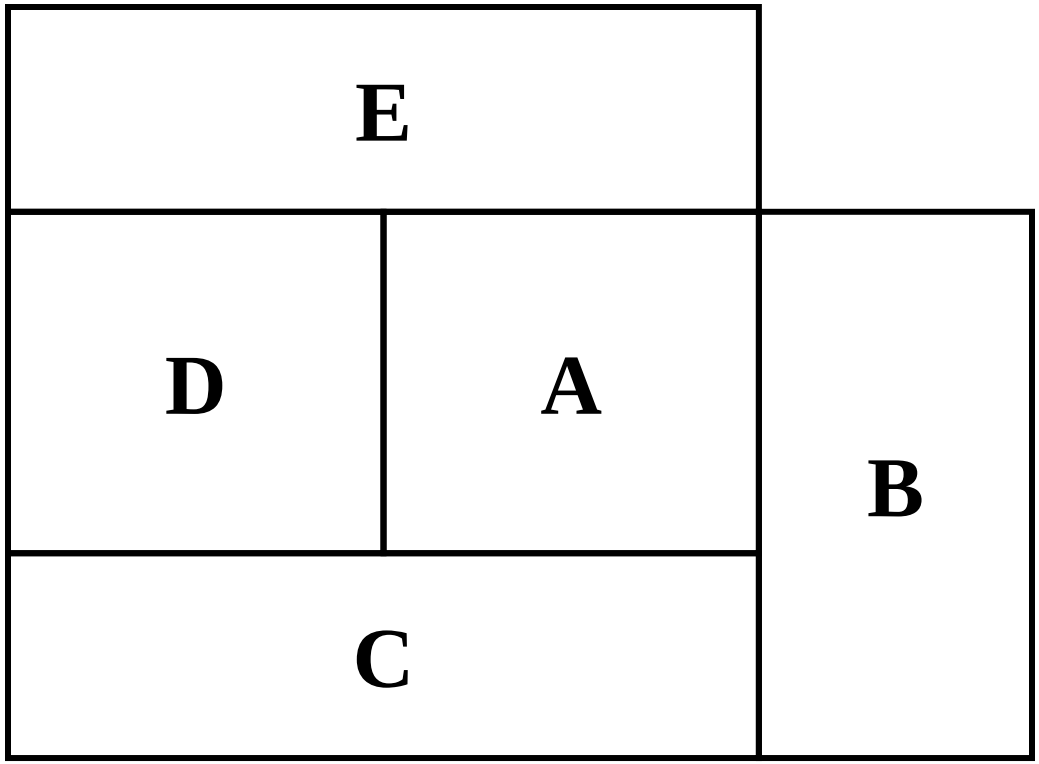


Examples of scheduling constraints:

- T1 must be done during T3
- T2 must be achieved before T1 starts
- T2 must overlap with T3
- T4 must start after T1 is complete

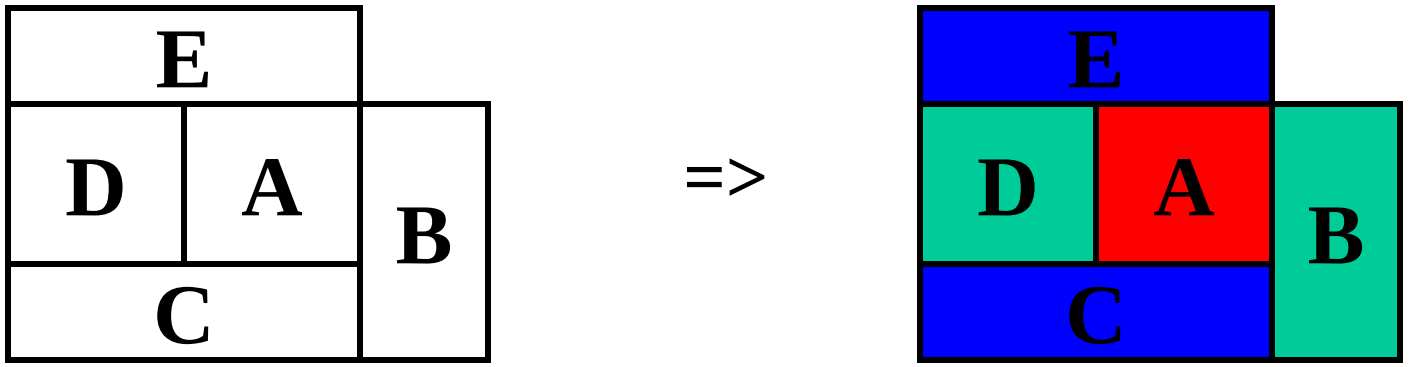
Example: Map coloring

Color the following map using three colors (red, green, blue) such that no two adjacent regions have the same color.



Map coloring

- Variables: A, B, C, D, E all of domain RGB
- Domains: RGB = {red, green, blue}
- Constraints: A≠B, A≠C,A ≠ E, A ≠ D, B ≠ C, C ≠ D, D ≠ E
- A solution: A=red, B=green, C=blue, D=green, E=blue



Brute Force methods

- Finding a solution by a brute force search is easy
  - **Generate and test** is a weak method
  - Just generate potential combinations and test each
- Potentially very inefficient
  - With n variables where each can have one of 3 values, there are 3^n possible solutions to check
- There are ~190 countries in the world, which we can color using four colors
- 4^{190} is a big number!

```
solve(A,B,C,D,E) :-
  color(A),
  color(B),
  color(C),
  color(D),
  color(E),
  not(A=B),
  not(A=B),
  not(B=C),
  not(A=C),
  not(C=D),
  not(A=E),
  not(C=D).

color(red).
color(green).
color(blue).
```

4**190 is 2462625387274654950767440006258975862817483704404090416746768337765357610718575663213391640930307227550414249394176L

Example: SATisfiability

- Given a set of propositions containing variables, find an assignment of the variables to {false, true} that satisfies them.
- For example, the clauses:
  - $(A \vee B \vee \neg C) \wedge (\neg A \vee D)$
  - (equivalent to $(C \rightarrow A) \vee (B \wedge D \rightarrow A)$

  are satisfied by

  A = false, B = true, C = false, D = false
- Satisfiability is known to be NP-complete, so in the worst case, solving CSP problems requires exponential time

Real-world problems

CSPs are a good match for many practical problems that arise in the real world

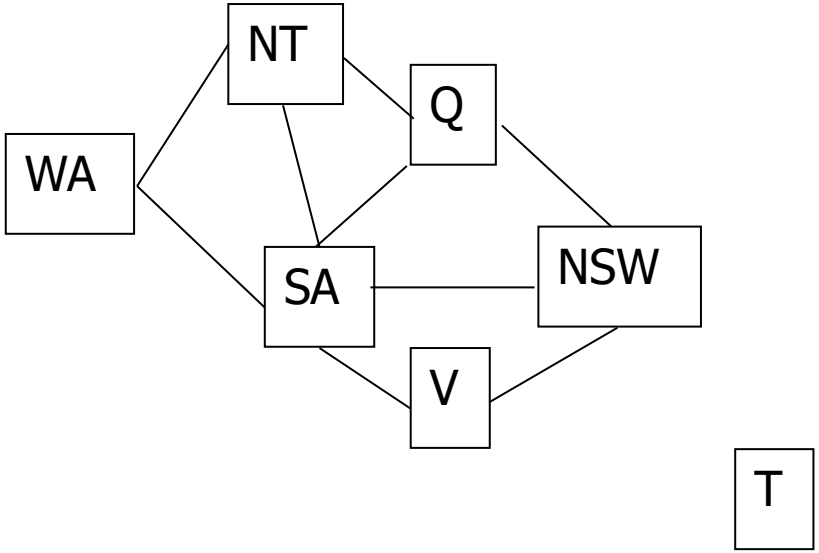
- Scheduling
- Temporal reasoning
- Building design
- Planning
- Optimization/satisfaction
- Vision
- Graph layout
- Network management
- Natural language processing
- Molecular biology / genomics
- VLSI design

Definition of a constraint network (CN)

A constraint network (CN) consists of

- a set of **variables** $X = \{x_1, x_2, \ldots x_n\}$
  - each with associated domain of values $\{d_1, d_2, \ldots d_n\}$
  - domains are typically finite
- a set of **constraints** $\{c_1, c_2 \ldots c_m\}$ where
  - each defines a predicate which is a relation over a particular subset of variables (X)
  - e.g., C_i involves variables $\{X_{i1}, X_{i2}, \ldots X_{ik}\}$ and defines the relation $R_i \subseteq D_{i1} \times D_{i2} \times \ldots D_{ik}$

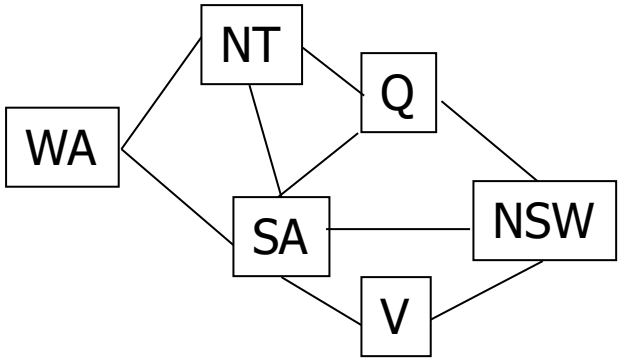
Running example: coloring Australia

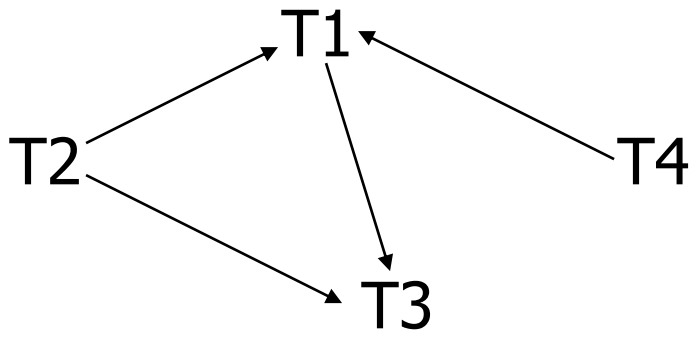


- Seven variables: **{WA,NT,SA,Q,NSW,V,T}**
- Each variable has the same domain: **{red, green, blue}**
- No two adjacent variables have the same value:
  WA≠NT, WA≠SA, NT≠SA, NT≠Q, SA≠Q, SA≠NSW, SA≠V,Q≠NSW, NSW≠V

Unary & binary constraints most common

Binary constraints





T

- Two variables are adjacent or neighbors if connected by an edge or an arc
- Possible to rewrite problems with higher-order constraints as ones with just binary constraints
  - Reification

Formal definition of a CN

- Instantiations
  - An **instantiation** of a subset of variables S is an assignment of a value in its domain to each variable in S
  - An instantiation is **legal** iff it does not violate any constraints
- A **solution** is an instantiation of all of the variables in the network

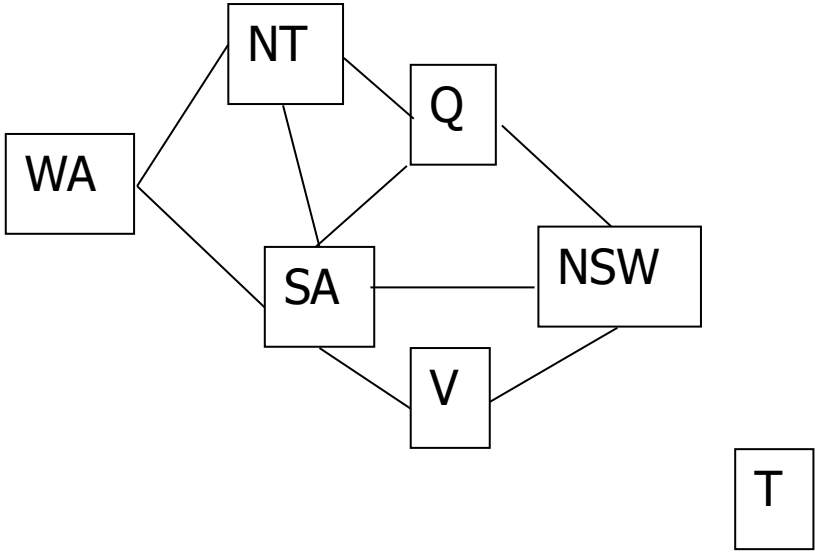
Typical tasks for CSP

- Solution related tasks:
  - Does a solution *exist*?
  - Find *one* solution
  - Find *all* solutions
  - Given a metric on solutions, find the *best* one
  - Given a partial instantiation, do any of the above
- Transform the CN into an equivalent CN that is easier to solve

Binary CSP

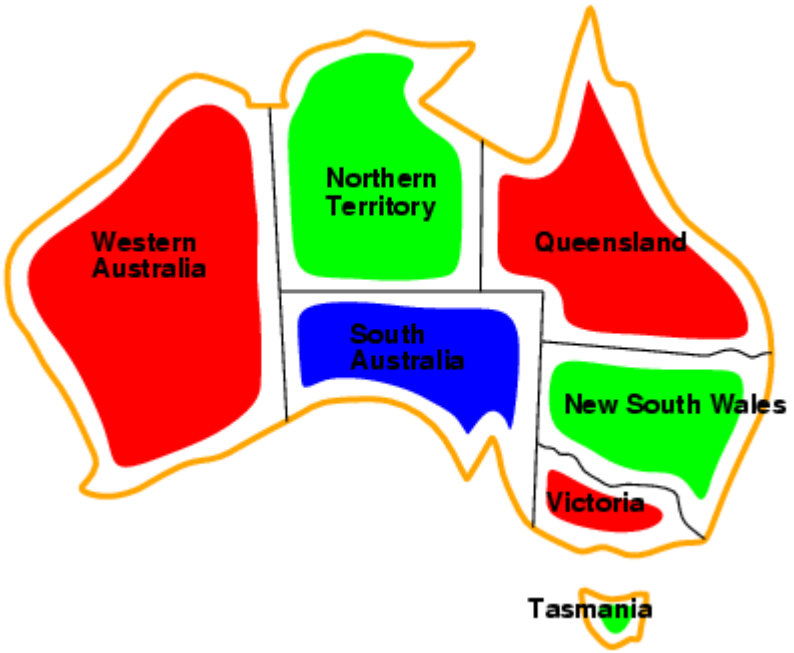
- A **binary CSP** is a CSP where all constraints are binary or unary
- Any non-binary CSP can be converted into a binary CSP by introducing additional variables
- A binary CSP can be represented as a **constraint graph**, with a node for each variable and an arc between two nodes iff there's a constraint involving the two variables
  - Unary constraints appear as self-referential arcs

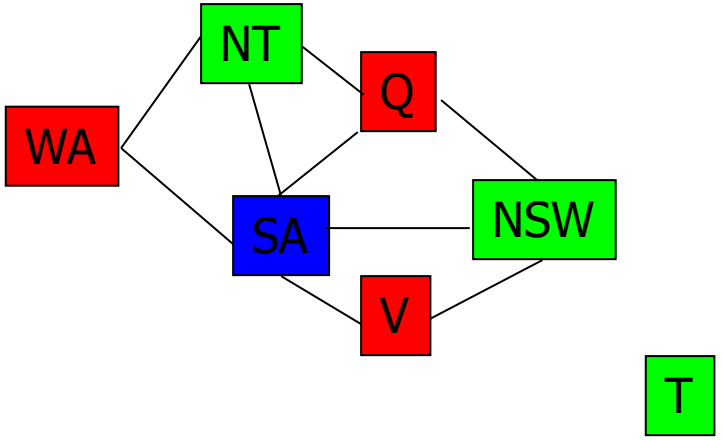
Running example: coloring Australia



- Seven variables: **{WA,NT,SA,Q,NSW,V,T}**
- Each variable has the same domain: **{red, green, blue}**
- No two adjacent variables have the same value:
  WA≠NT, WA≠SA, NT≠SA, NT≠Q, SA≠Q, SA≠NSW, SA≠V,Q≠NSW, NSW≠V

A running example: coloring Australia





- Solutions are complete and consistent assignments
- One of several solutions
- Note that for generality, constraints can be expressed as relations, e.g., WA ≠ NT is

  (WA,NT) in {(red,green), (red,blue), (green,red), (green,blue), (blue,red),(blue,green)}

Challenges for constraint reasoning

- What if not all constraints can be satisfied?
  - Hard vs. soft constraints
  - Degree of constraint satisfaction
  - Cost of violating constraints
- What if constraints are of different forms?
  - Symbolic constraints
  - Numerical constraints [constraint solving]
  - Temporal constraints
  - Mixed constraints

Challenges for constraint reasoning

- What if constraints are represented intensionally?
  - Cost of evaluating constraints (time, memory, resources)
- What if constraints, variables, and/or values change over time?
  - Dynamic constraint networks
  - Temporal constraint networks
  - Constraint repair
- What if multiple agents or systems are involved in constraint satisfaction?
  - Distributed CSPs
  - Localization techniques

Metareasoning

- Metareasoning is a general AI term for “thinking about thinking” within a computational system.

- While reasoning algorithms are used to make decisions, a metareasoning algorithm is used to control a reasoning algorithm or to select among a set of reasoning algorithms, determining which decision-making method should be used under different circumstances.

- A classic example of metareasoning is to determine whether a reasoning algorithm should stop or continue in a given context.

- Metareasoning can be described as in Fig. 1, where reasoning occurs at the Object Level based on observations at the Ground Level, and the decisions made at the Object Level are enacted at the Ground Level.

- For example, a sensored alarm might sound at the Ground Level when an algorithm at the Object Level determines from sensor input that an intruder was present (e.g., this algorithm may sound an alarm when two or more motion events are detected within a 10-s window).

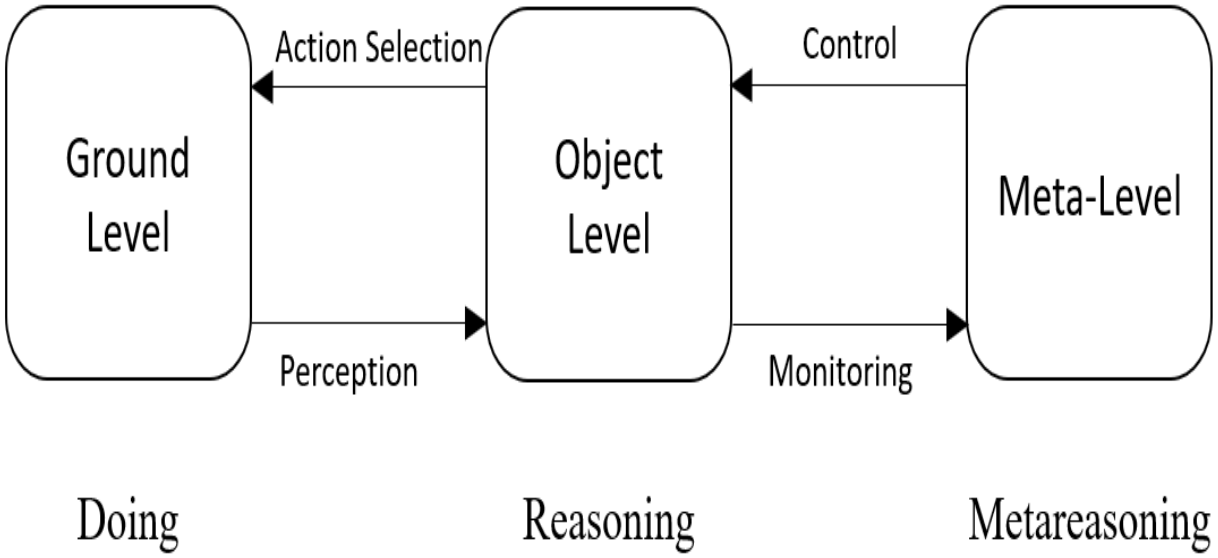


Fig.1 Classic decision–action loop diagram of metareasoning, where reasoning happens at the object level to select the actions that will happen at the ground level, and metareasoning happens at the meta-level to control what occurs at the object level

- Metareasoning then occurs when information from the Object Level is observed and altered at the Meta-Level.

- In the previous example, an algorithm at the Meta-Level might adjust the sensitivity of the alarm if it is triggered too often, causing battery issues (e.g., this Meta-Level algorithm might impose a new algorithm at the Object Level that sounds the alarm only when three or more motion events are detected within a 10-s window).

Metareasoning within a single agent and multi-agent

- Metareasoning can occur within a single agent, as depicted in Fig. 1, or it can occur within a multi-agent system (MAS) as you can see in figure (2-a).

- Metareasoning is often used in a multi-agent setting to optimize the performance of an entire system, and there are many options for how it is implemented with different consequences for resources such as time and compute power.

- For example, agents within an MAS may perform their metareasoning independently and communicate at the Object Level, which may be a good solution when communication is costly and coordination is a low priority.

- When coordination is more important, independently metareasoning agents may communicate at the Meta-Level to jointly determine how they will independently metareason.

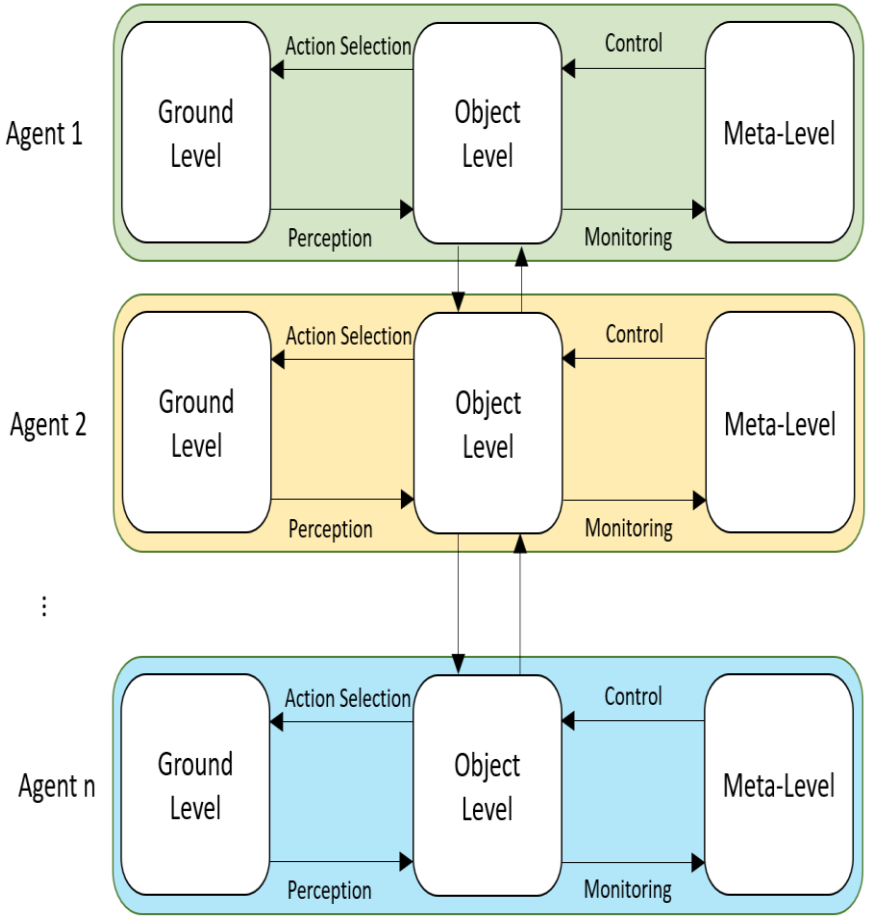


Fig.2(a) An MAS system where metareasoning occurs independently for each agent.

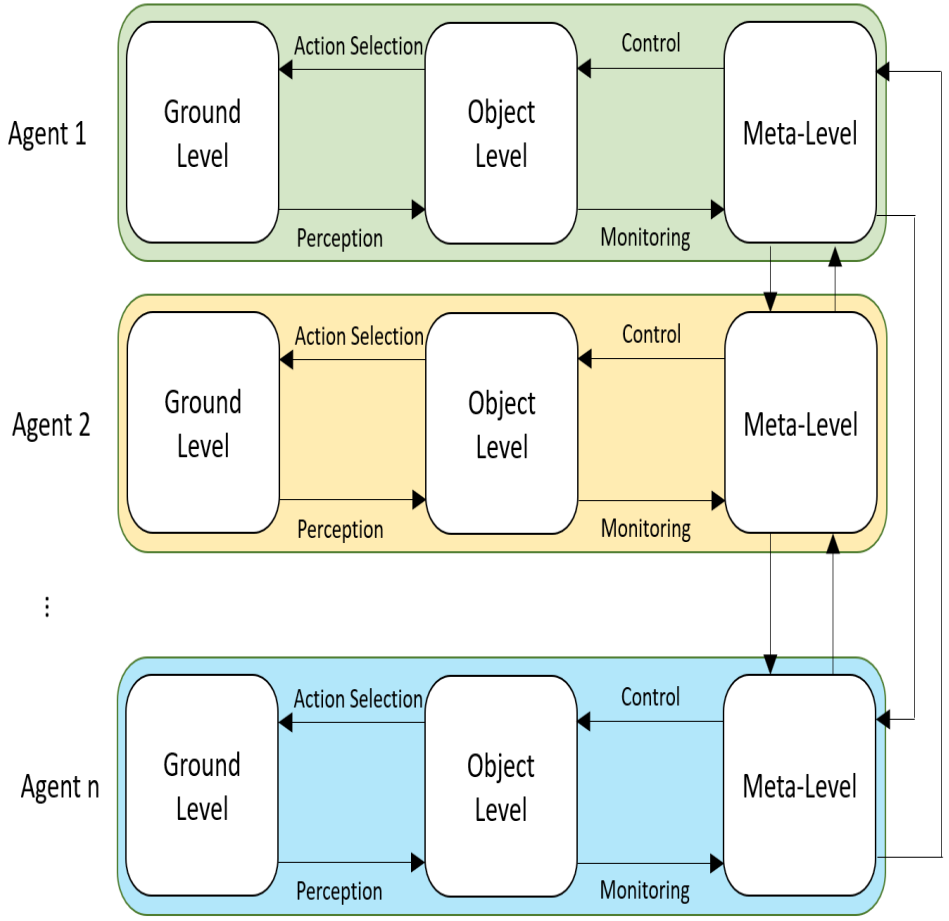


Fig.2(b) An MAS where each agent's metareasoning communicates and coordinates with each other agent's metareasoning.

An MAS with a single centralized and multiple separate metareasoning agents

- Metareasoning can also be performed in a more centralized fashion by separate metareasoning agents (Fig.3-a). As communication resources allow, the best coordination and metareasoning is expected to come from a single centralized metareasoning agent (Fig. 3-b).

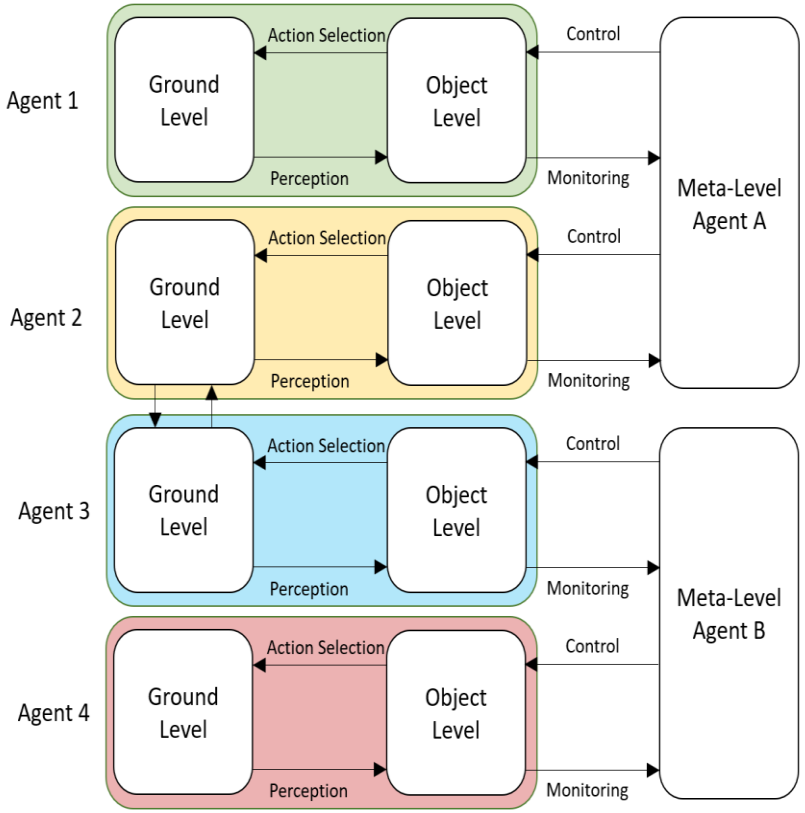


Fig.3-a An MAS with multiple separate metareasoning agents

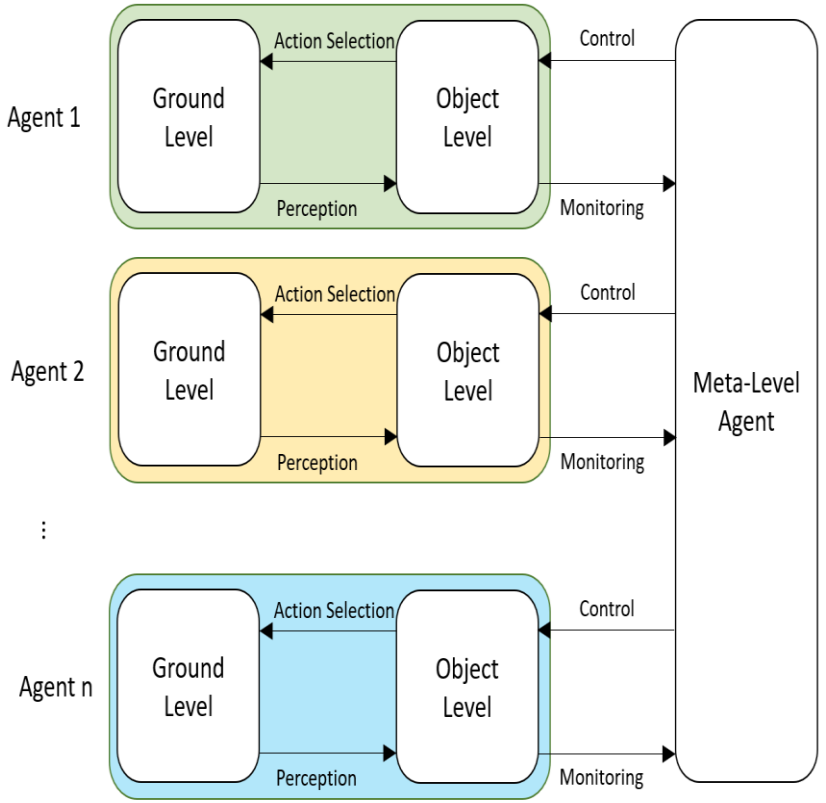


Fig.3-b An MAS with a single centralized metareasoning agent.

- Systems also vary in the object of their metareasoning. Single agent metareasoning is often used to control algorithm halting or switching and applied to a wide variety of fields, including scheduling and planning, heuristic search, and object detection.

- Within MASs, metareasoning is often used to control communication and resources within the systems, including controlling communication frequency or content, or assigning tasks.

An additional concern in metareasoning is how much learning or metareasoning should happen online versus offline. Because online metareasoning can be costly in terms of time and computation, offline policies are often maximized to the extent that they do not unduly impair system accuracy.

Broadly speaking, ToM (Theory of Mind) is a form of metareasoning, or "thinking about thinking." As described in Fig. 1, however, metareasoning is performed through monitoring and controlling the Object Level, whereas ToM involves making inferences from what is happening at the Ground Level without direct access to the Object Level (e.g., an agent's beliefs).

- While metareasoning is already widely used in single- and multi-agent systems to improve performance, ToM approaches arguably have not been explored as deeply as a method to improve performance of an artificially intelligent agent.

- This is almost certainly in part because ToM is more closely tied to human cognition, which places strong restrictions on plausible ToM models and biases research toward human applications.

- Additionally, ToM itself is still somewhat controversial (e.g., Who has it? When is it acquired? Under what conditions is it exercised?) but it holds promise for creating more-transparent (if not authentically human) systems, especially systems reasoning with multiple sources of information and with differing provenance and certainty.

- In particular, recent computational ToM approaches, which use simpler, heuristic definitions of ToM, may be the best source of innovation in this field.