Artificial Intelligence

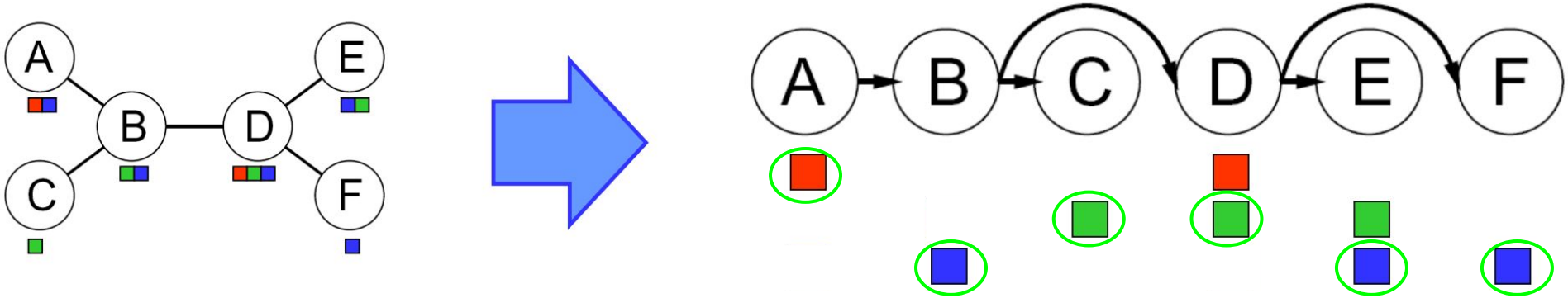
Lec 15: CSP (contd.), Knowledge Representation and Logic

Pratik Mazumder

Content taken from various sources

Tree-Structured CSPs

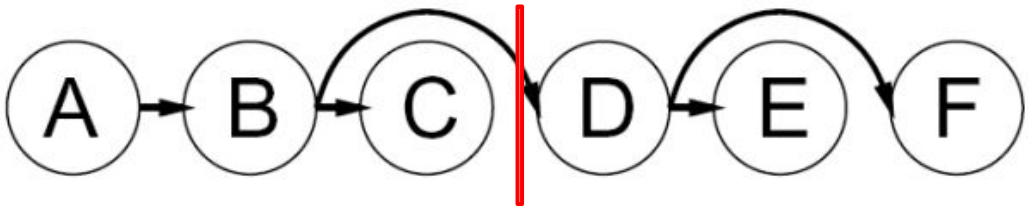
- Algorithm for tree-structured CSPs:
  - Order: Choose a root variable and order variables so that parents precede children.



  - Remove backward: For i = n to 2, apply RemoveInconsistent(Parent(X_i),X_i)
    - One backward pass from F back to A and enforce arc consistency
  - Assign forward: For i = 1 to n, assign X_i consistently with Parent(X_i)
    - One forward pass from A to F and assign one of the available colors consistent with the parent

Tree-Structured CSPs

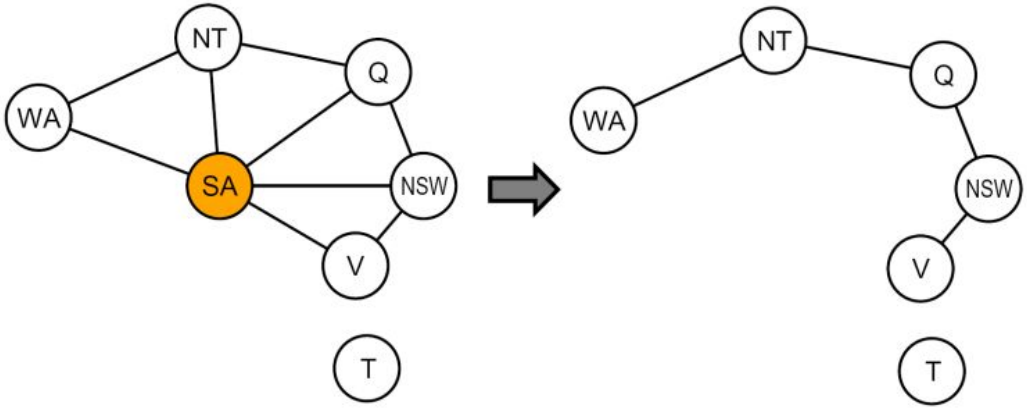
- **After backward pass, all root-to-leaf arcs are consistent.**
- Justification: Each X->Y was made consistent at one point, and Y's domain could not have been reduced thereafter (because Y's children were processed before Y).
  - During the backward pass, we remove the conflicting colors from the **parent** to avoid conflict with the **children**.
  - So once we move further backward, the domain of the previous parent node can no longer be changed.
    - since this is a tree and all children nodes are already visited



- **If root-to-leaf arcs are consistent, forward assignment will not backtrack**
- Justification: Because we have already handled all possible cases of conflict during the backward pass.
  - Therefore, for whatever color we choose for the parent, there will be a non-conflicting color at the children nodes and so on.

Improving Structure: Nearly Tree-Structured CSPs

- Not all CSPs will be Tree-Structured.
- Sometimes deleting a variable/node will make the problem tree-structured.

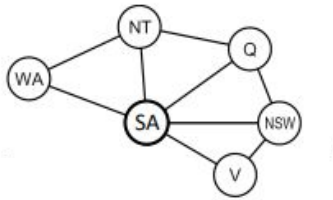


- But we cannot delete, so let's do the next best thing.
- Conditioning: **instantiate a variable**, **prune** its **neighbors' domains**
  - For all further processing, the instantiated variable is a constant and does not need to be handled

- Cutset conditioning: instantiate (in all ways) a set of variables such that the remaining constraint graph is a tree (cycle cutset).

Cutset Conditioning

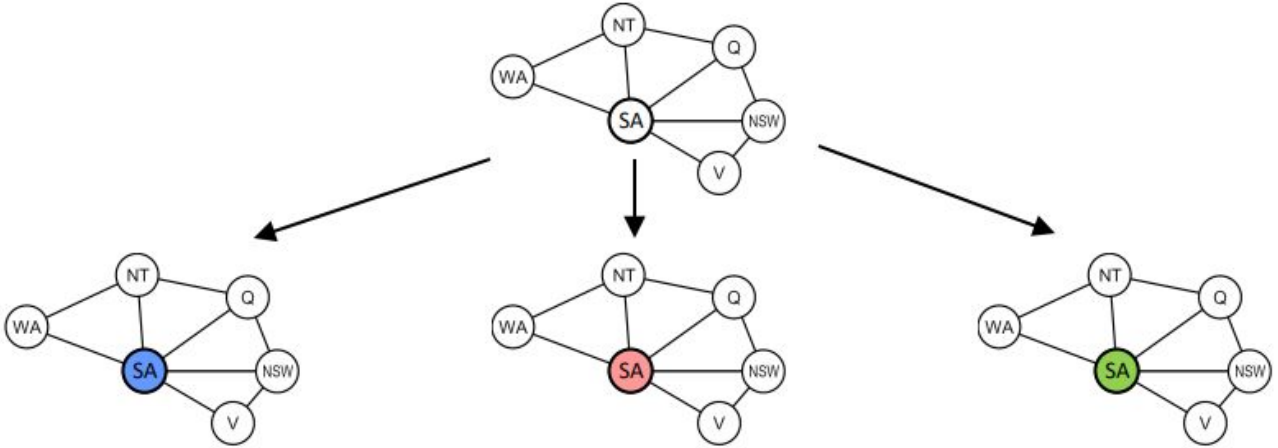
Choose a cutset



Cutset Conditioning

Choose a cutset

Instantiate the cutset (all possible ways)



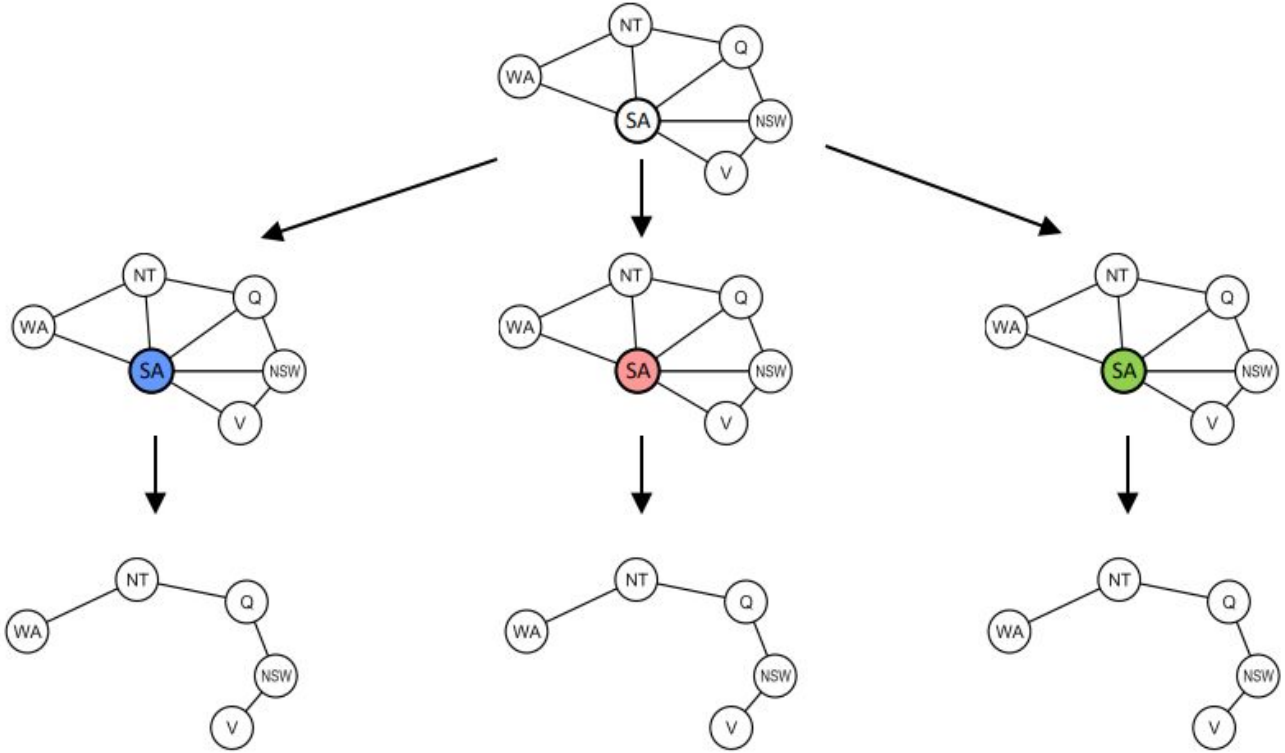
Cutset Conditioning

Choose a cutset

Instantiate the cutset (all possible ways)

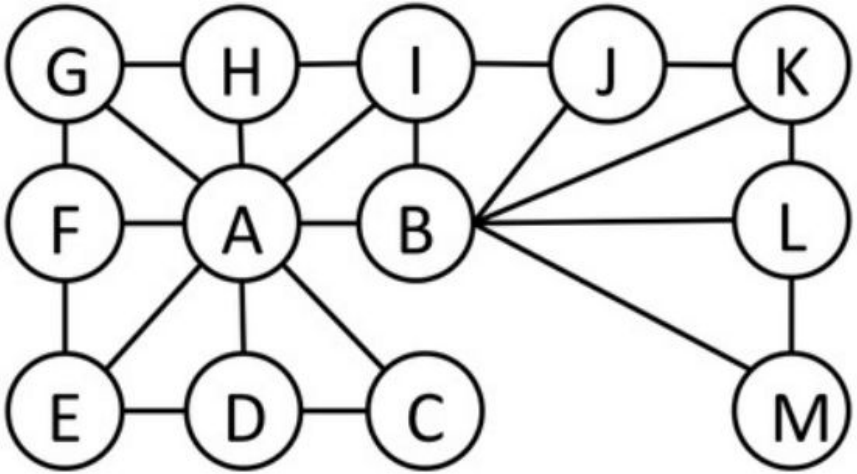
Compute residual CSP for each assignment

Solve the residual CSPs (tree structured)



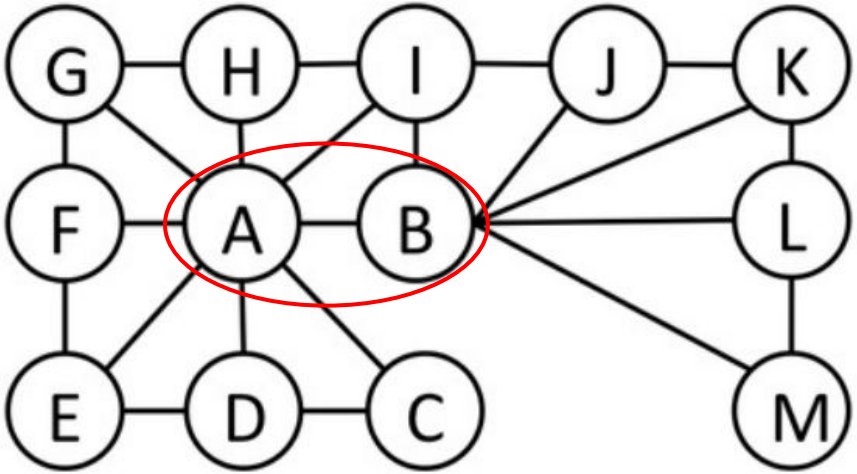
Cutset Conditioning

Find the smallest cutset to make it a tree.



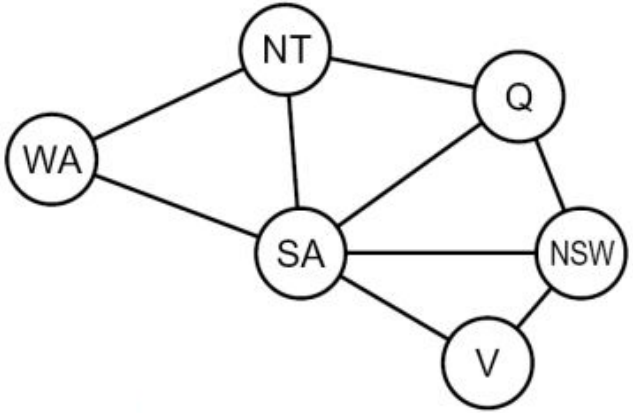
Cutset Conditioning

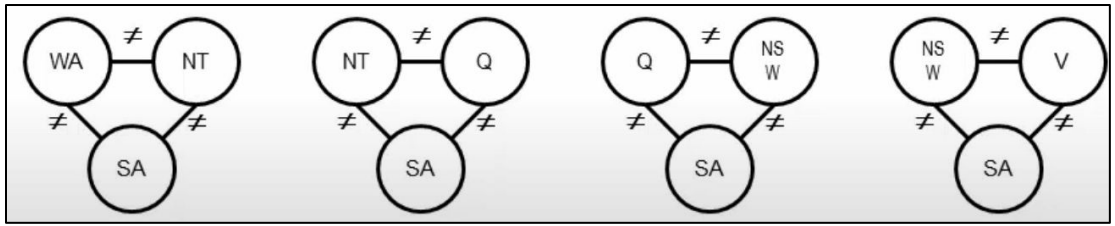
Find the smallest cutset to make it a tree.



Tree Decomposition

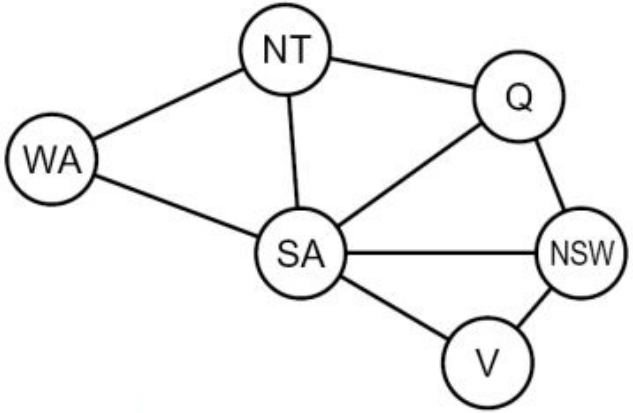
- Idea: create a tree-structured graph of **mega-variables**.
- Each mega-variable encodes part of the original CSP.
- Subproblems overlap to ensure consistent solutions,
  - e.g., same color for SA in M1 and M2.
- May not be always possible if densely connected graph

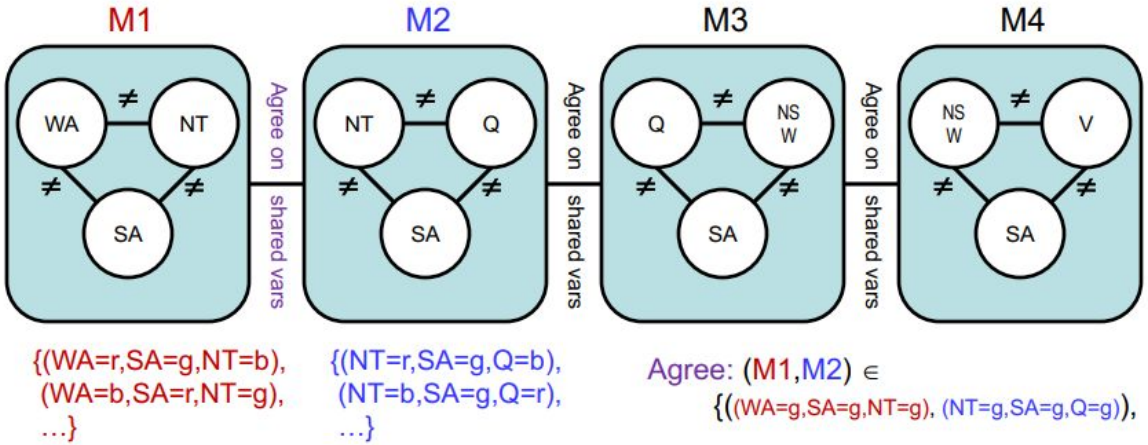




Tree Decomposition

- Idea: create a tree-structured graph of **mega-variables**.
- Each mega-variable encodes part of the original CSP.
- Subproblems overlap to ensure consistent solutions,
  - e.g., same color for SA in M1 and M2.
- May not be always possible if densely connected graph





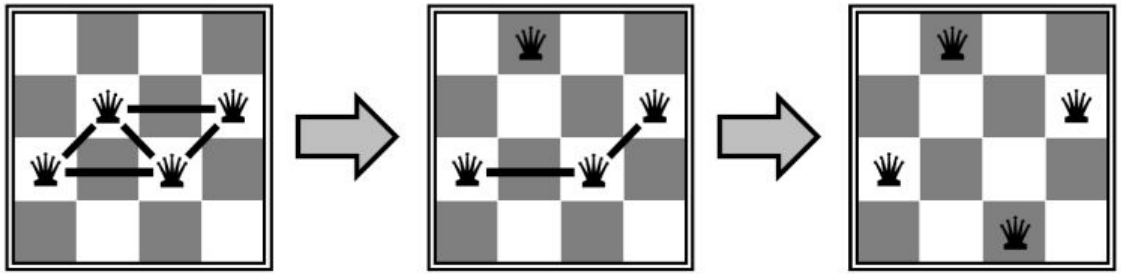
Iterative Algorithms for CSPs

- Local search methods typically work with “complete” states, i.e., all variables assigned
  - Basically, start with a full assignment that may be inconsistent and iteratively correct it.

- To apply to CSPs:
  - Take an assignment with unsatisfied constraints.
  - Operators reassign variable values.

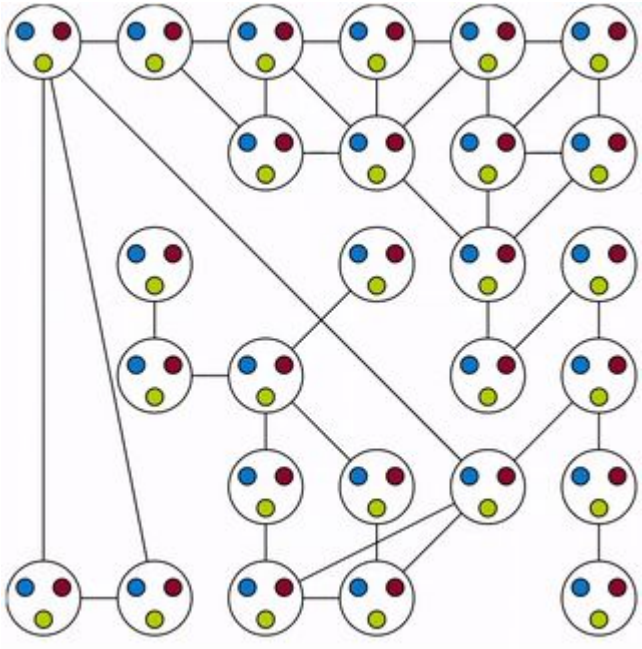
- Algorithm:
  - Variable selection: randomly select any conflicted variable.
  - Value selection: **min-conflicts** heuristic:
    - Choose a value that violates the fewest constraints.
    - i.e., hill climb with h(n) = total number of violated constraints.

Example: 4-Queens

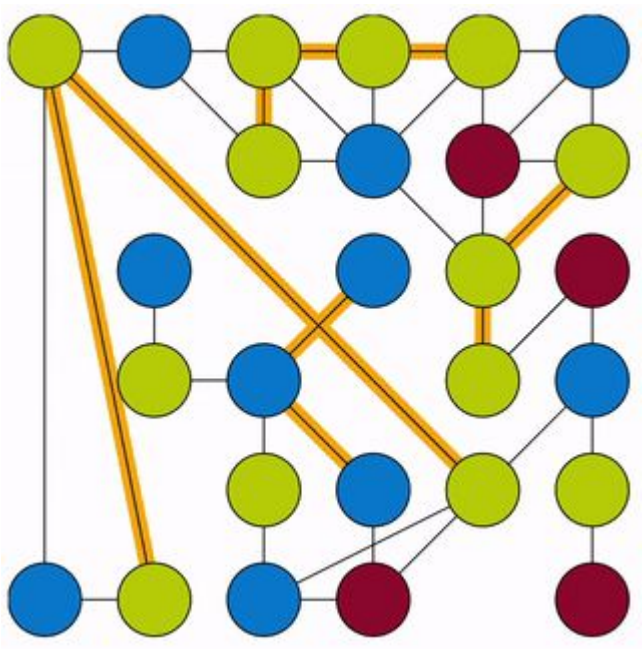


- States: 4 queens in 4 columns ($4^4 = 256$ states)
- Operators: move queen in column
- Goal test: no attacks
- Evaluation: c(n) = number of attacks

Iterative Algorithms for CSPs



Backtracking with Arc Consistency with MRV and LCV



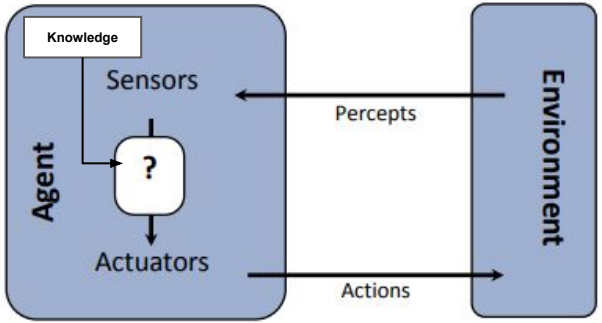
Iterative Algorithm

Performance of Min-Conflicts

- Given random initial state, can solve n-queens in almost constant time for arbitrary n with high probability.
- It solves even the million-queens problem in an average of 50 steps (after the initial assignment).

Knowledge and Intelligence

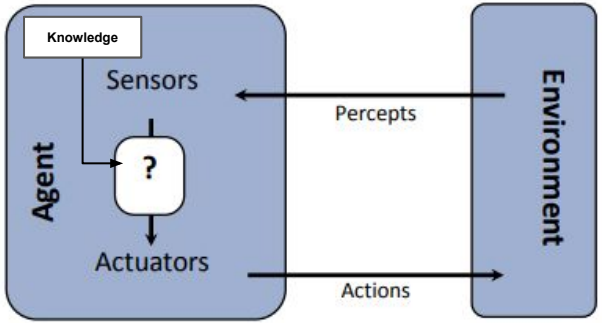
- What is the role that knowledge plays in intelligent behavior?



- The agent senses some event in the environment.
- Based on the precepts sensed by the agent, the agent makes some decisions.
- The agent takes the help of knowledge to decide on what actions to take.
- The decisions make the agent take some actions.

Knowledge and Intelligence

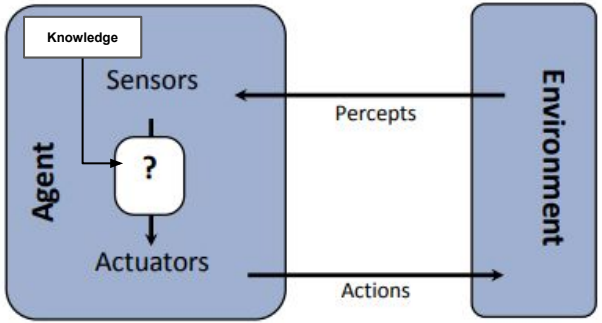
- What is the role that knowledge plays in intelligent behavior?



- If you remove knowledge, the agent will not be able to make a proper decision about which action to take.
- Therefore, knowledge helps the agent act intelligently in a particular scenario in the environment.

Knowledge and Intelligence

- What is the role that knowledge plays in intelligent behavior?



- We want to build agents that behave like human beings or at least demonstrate some intelligent behavior.
- In order to build such an agent, we need to have a way to represent knowledge such that an agent can understand it.

Knowledge Representation and Logic

- Till now we have looked at search problems and different searching techniques to solve these problems.
- One of the most important aspects of any artificial intelligence activity is how to represent knowledge, that is, knowledge representation.
- One of the major techniques of knowledge representation is logic.
- We will be discussing the different knowledge representation schemes.
    - Represent knowledge about the world in a manner that facilitates inferencing (i.e., drawing conclusions) from knowledge.
    - Example: Arithmetic logic
      x>=5

Knowledge Representation and Logic

- We need a language to represent domain knowledge.
    - There must be a method to use this knowledge.
    - The method will be used to interpret knowledge in response to any event occurring in the environment.

Inference Mechanism

- An **inference** mechanism **takes** the **precepts** sensed from the environment, **interprets** the knowledge that has been represented in a suitable language, and uses that knowledge to **act** according to the need.
- Therefore, it can generate proper actions, which helps the agent behave intelligently.
- In order to demonstrate intelligent behavior, the agent needs the knowledge represented in some language and the method to use the language to interpret the knowledge.
- **Knowledge representation** should **include both**: the **language to represent knowledge** and an **inference mechanism that can use this knowledge**.

Knowledge Representation and Logic

- Logic is one such formal knowledge.
- It provides a formal system for evaluating the validity of arguments and drawing conclusions from a set of premises.
- In the context of artificial intelligence, logic is used as a tool for representing and reasoning about knowledge.
- There are several types of logic, including propositional logic, first-order logic, modal logic, and many others.
- Each type of logic has its own set of rules and symbols for representing knowledge and drawing inferences.
- An important tool for building intelligent agents that can reason about the world and make decisions based on that reasoning.

Knowledge Representation and Logic

- The central component of a knowledge-based agent is its **knowledge base**.
- A knowledge base is a set of **sentences**.
- Each sentence is expressed according to the syntax of the knowledge representation language (logic in this case) and denotes some assertion about the world.
  - x + y = 4 vs x4y+=
- A logic must also define the semantics or meaning of a sentence.
- The semantics defines the truth of each sentence with respect to each possible world.
- e.g., the semantics for arithmetic specifies that the sentence “x + y = 4” is true in a world where x is 2 and y is 2, but false in a world where x is 1 and y is 1.
- In standard logic, every sentence must be either **true** or **false** in each possible world.

Knowledge Representation and Logic

- A logic includes
  - Syntax: What is a correctly formed sentence?
  - Semantics: What is the meaning of the sentence?
  - Inference Procedure (reasoning, entailment): what sentence logically follows from the given knowledge?

Propositional Logic (PL)

- Simple but powerful logic
- Building blocks of PL are propositions.
- A proposition is a statement that is either true or false.
- Atomic sentences consist of a single proposition symbol.
- A symbol in PL is a symbolic variable that stands for a proposition that can be true or false.
- Complex sentences are constructed from simpler sentences, using parentheses and logical connectives.

Propositional Logic (PL)

- There are five connectives in common use:
  - NEGATION ¬ or ! (not): ¬P or !P is called the negation of P.
    - A literal is either an atomic sentence (a positive literal) or a negated atomic sentence (a negative literal).
  - CONJUNCTION ∧ (and): A sentence whose main connective is ∧, such as P ∧ Q, is called a conjunction
    - its parts are the **conjuncts**.
  - DISJUNCTION ∨ (or): A sentence using ∨, such as (P ∧ Q) ∨ R, is a disjunction of the **disjuncts** (P ∧ Q) and R.
  - IMPLICATION ⇒ (implies). A sentence such as (P ∧ Q) ⇒ R is called an implication (or conditional).
    - Its premise or antecedent is (P ∧ Q), and its conclusion or consequent is R.
  - BICONDITIONAL ⇔ (if and only if). The sentence P ⇔ ¬Q is a biconditional.