Artificial Intelligence (Logic and Reasoning Patterns, and State Space Search)

Prof K R Chowdhary

Formerly, at CSE Dept., MBM Engg. College

January 27, 2025
Lecture #2

Predicate Logic

Blocks world. Physical objects, like - *cuboid*, *cone*, *cylinder* placed on the table-top, with some relative positions, as shown in figure 1. Prog. file: (blocks.pl) [2].

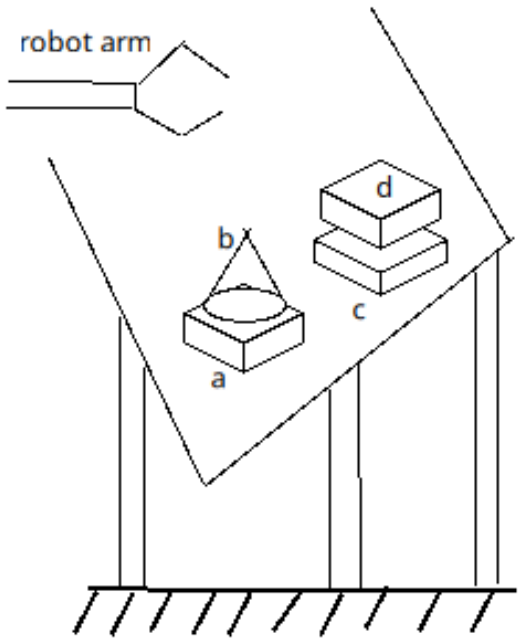

Figure 1: Blocks World

```
cone(b). cuboid(a).
cuboid(c). cuboid(d).
onground(a).
onground(c).
ontop(b,a).
ontop(d,c).
topclear(b).
topclear(d).
puton(X, Y) :- topclear(X),
 topclear(Y), not(cone(Y)),
 not(X=Y).
```

Run it using swi Prolog:

$ swipl

[blocks].

?

Rule based system (RBS)

Rule Based Reasoning: A rule is in the form of "*if-then*" or "$(p \rightarrow q)$," where p is called precondition or premises, or assertion and q is called consequence, goal or hypothesis [1], [3].

• Inference is carried out for chained rules: *forward-chaining* and *back-ward* chaining.

• Rule based system (RBS) consists of a knowledge base and an inference engine (figure 2). Knowledge base = "rules and facts." Rules express a conditional, with a precondition and a consequent.

• Interpretation of a rule: "if the precondition can be satisfied, the consequent holds."

• Rules express deductive knowledge, e.g., logical relationships. This supports inference (verification, or evaluation) tasks.

• Rules express goal-oriented knowledge that RBS can apply in seeking problem solutions.

Rule based system (RBS) ...

- Rules express causal relationships, which RBS uses to answer "what if" questions, or to determine possible causes for specified events.
- In RBS, each *if* pattern may match to one or more of *assertions* in a collections of assertions in working-memory, showwn in fig. 2. Resulting "*Then*" patterns are assertions to be placed into working memory.

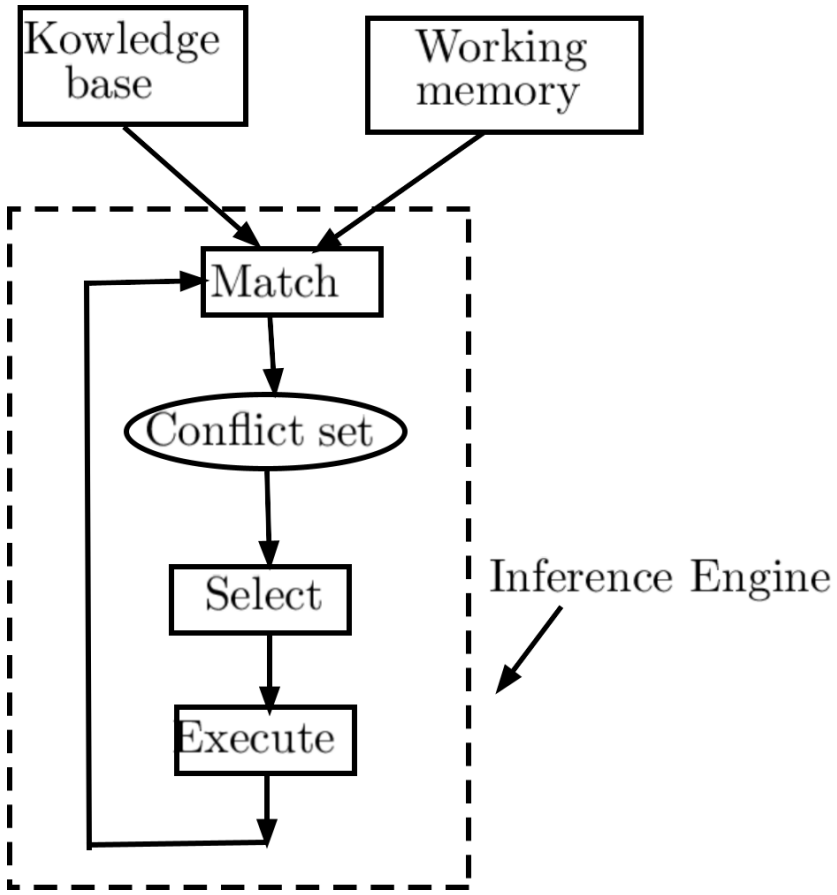

Figure 2: Inference Cycle.

Rule based system ...

Facts - Assertions is knowledge about properties, relations, propositions. Rules, which RBS interprets as imperatives, facts are static and inactive.

A system like this is called *deduction system*. Rule:

```
Rule: if cond.1
          if cond.2
             ...
          then    ...
        then    ....
```

R_i rule's name or number. Assertions, i.e., working memory, matches with one or more "if" patterns, like if1, if2, and produces new assertions.

- First expert system was MYCIN: Demonstrated potential of AI in practical applications, performance close to humans. MYCIN's[1] rule:

```
R1:if stiff neck and
       high temperature
       impairment of consciousn
                  occur together,
 then
  meningitis is suspected.
```

[1] meningitis= "Inflation of membrane of spinal chord and brain."

Introduction to search in AI

State-Space: • State space consists set of vertices V and set of connections between them as edges (links), E. Thus, the search space is a graph $G = (V, E)$. The states are visited once only in process of search, search paths takes shape of a tree, and not graph [4].

• In conventional trees and graphs the space limits are fixed, i.e., number of vertices, and edges connecting them are already given. However. In AI search problems, alternate moves are generated from each vertex, like in a chess game, search tree is to be generated and simultaneously be searched.

• Initial configuration of the problem description is *start* state, at which we apply the specified rules to generate new states (vertices).

• Thus, it becomes similar to a tree with start configuration as *root* node, then there are interior nodes, and child nodes at the lowest level having no further children.

Search in AI ...

- The 8-puzzle game is shown in figure 3, with initial configuration, goal configuration, and transitions (moves) possible from each state.
- We refer the configuration of the game equal to the collective status of tiles on a board. This configuration we call as state.
- What are the three things of PSS, here in: Symbols, manipulation, and search?

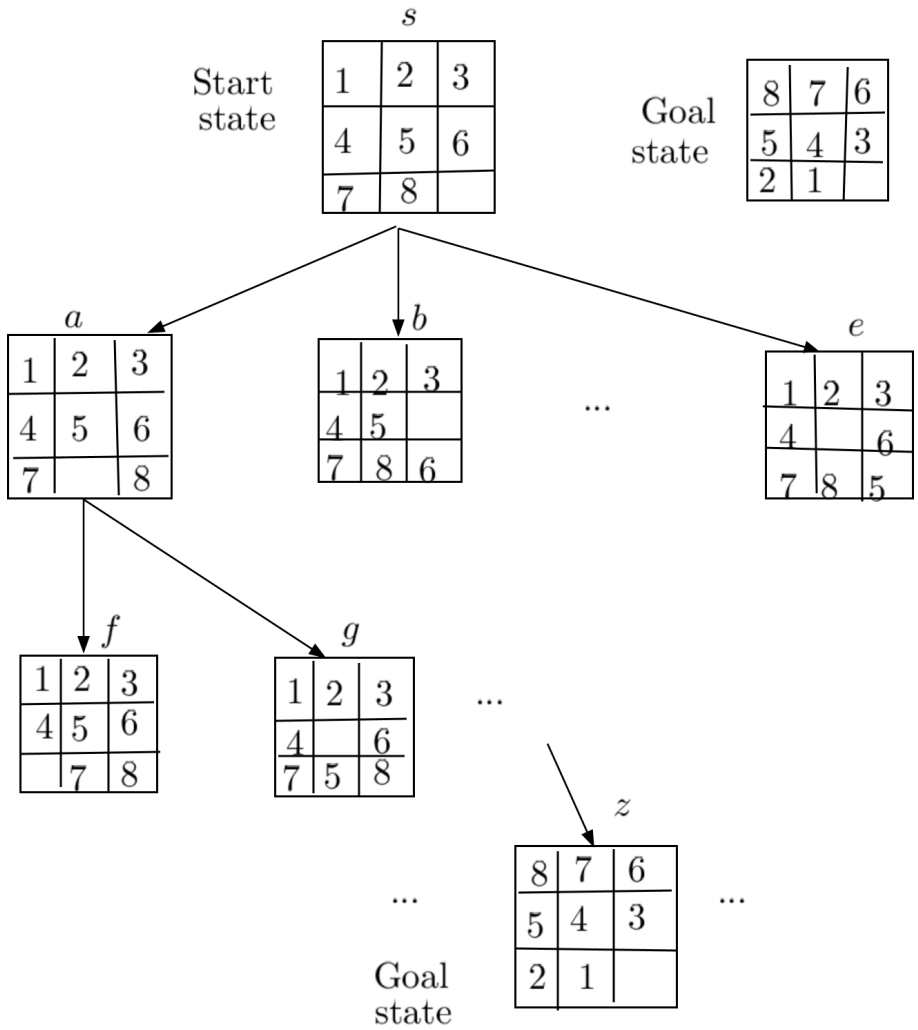

Figure 3: The 8-puzzle Game.

State-space Search

- Let us try to find out number of states to be visited in the worst case to reach to goal state in the case of search required in figure 3.
- Once a state is visited in a search path, it should not be reached again in the future in that path, otherwise there will be a cycle formed and we never reach to goal state.
- All unique states generated (like shown in figure as $a, b, c, \ldots, i, \ldots$), in the path to *goal*, in worst is $9! = 362880$.

- Tree search can end-up repeatedly visiting same nodes, unless it keeps track of all nodes visited. This could take vast amounts of memory, that most computers do not have, even for a graph of 50 nodes.

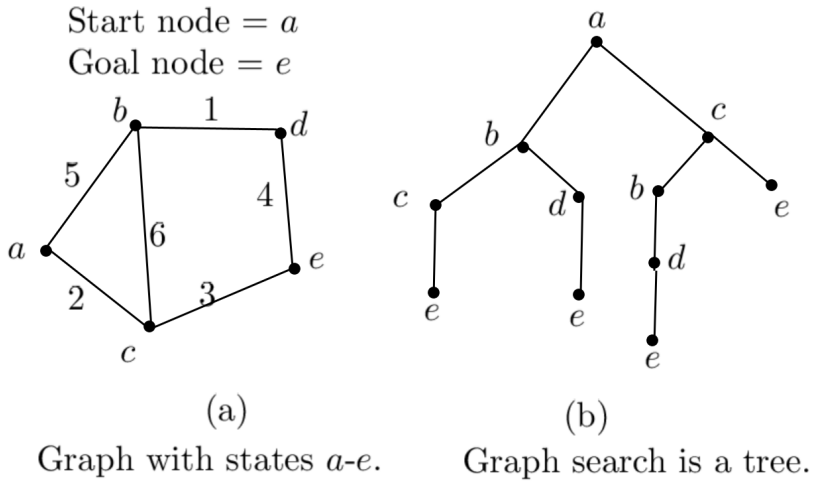

Figure 4: Undirected graph.

State-space Search...

• Since all the nodes are required to be searched - the search is called *exhaustive* or *blind* or *uninformed* search. Two important approaches are: *breadth-first search* (BFS) and *depth-first search* (DFS).

• Others approaches are their variants, for example, *depth-limited* search, *iterative deepening DFS*, and *bidirectional search*.

Breadth-First Search: A BFS phenomena is indicated in Fig 5 where dotted trace is showing order in which nodes are tested for the goal.

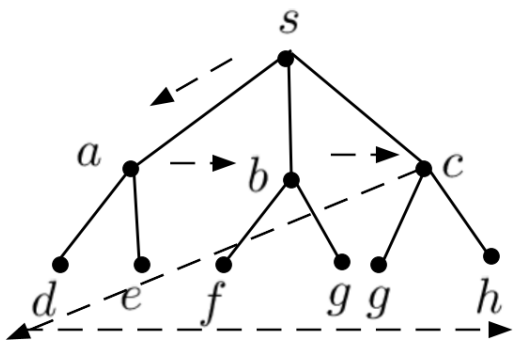

Figure 5: Breadth-first Search.

Breadth-First Search...

- *BFS* can be implemented using a *queue* type data structure: **List**. Front node of queue is represented by **List**.*Head*, Algorithm 1.
- When BFS algorithm 1 is applied for generate-and-search, a tree like one shown in Fig 5.
- The BFS order of the above case is $s, a, b, c, d, e, f, g, h$. But, path to the goal is not remembered!
- This is simple: In *queue* data structure each entry is stored as (x, y), where x is next child node, and y is parent.

Algorithm 1 BFS(Input: **G**, **S**, **Goal**)

```
1:  List.Head = [s]
2:  repeat
3:      if List.Head = Goal then
4:          return success
5:      end if
6:      generate children set C of List.Head
7:      append C to List
8:      delete List.Head
9:  until List = []
10: return fail
```

Logic and Reasoning Patterns References

[1] Chowdhary, K.R. (2020). Logic and Reasoning Patterns. In: Fundamentals of Artificial Intelligence. Springer, New Delhi. https://doi.org/10.1007/978-81-322-3972-7_2(pages: 25-50)

[2] Chowdhary, K.R. (2020). First Order Predicate Logic. In: Fundamentals of Artificial Intelligence. Springer, New Delhi. https://doi.org/10.1007/978-81-322-3972-7_3 (Pages: 51-88)

[3] Chowdhary, K.R. (2020). Rule Based Reasoning. In: Fundamentals of Artificial Intelligence. Springer, New Delhi. https://doi.org/10.1007/978-81-322-3972-7_4 (Pages: 89-109)

[4] Chowdhary, K.R. (2020). State Space Search. In: Fundamentals of Artificial Intelligence. Springer, New Delhi. https://doi.org/10.1007/978-81-322-3972-7_8(pages: 217-237)