Regular Expression (Regular Languages) FA.

→ Method to Represent a language (L)

$L = \{\lambda, a, aa, aaa, \ldots\}$ $\quad \lambda = \varepsilon$, a^*

→ Let 'R' be a Regular Expression over Alphabet Σ if R is:

1) λ 'ε', is Regular expression denoting the set $\{\varepsilon\}$
2) ϕ, is Regular expression denoting the empty set $\{\}$
3) For each symbol $a \in \Sigma$, a is regular expression denoting set $\{a\}$
4) Union of two RE is also Regular
5) Concatenation of two RE is also Regular
6) Kleene Closure * of RE is also Regular
7) if R is Regular, (R) is also Regular
8) Nothing else, Repeat 1 to 7 Recursively

$R = \varepsilon \quad L(R) = \{\varepsilon\}$

$R = \phi \quad L(R) = \{\}$

$R = a \quad L(R) = \{a\}$

$R_1 \cup R_2$ = Regular

$R_1 \cdot R_2$ = Regular

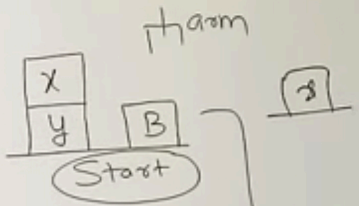
$a \cup b = \{a, b\}$

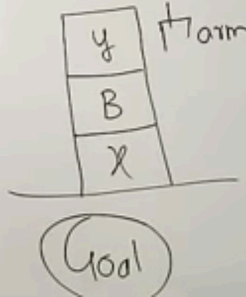
a^* = Reg. $\{\varepsilon, a, aa, aaa \ldots\}$

$(RE)^*$ = Regular

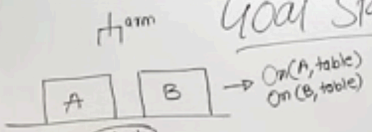
Goal Stack Planning

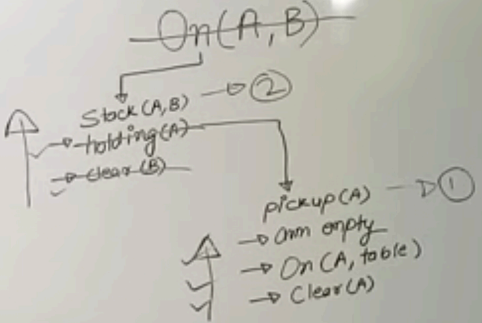
	Precondn	action
① Pickup(x)	• arm empty • on(x, table) • clear(x)	• holding(x)
② putdown(x)	• holding(x)	• arm empty • on(x, table) • clear(x)
③ Stack(x, y)	• holding(x)	• On(x, y) • Clear(x) • Arm empty
④ Unstack(x, y)	• on(x, y) • Clear(x) • arm empty	• holding(x) • Clear(y)

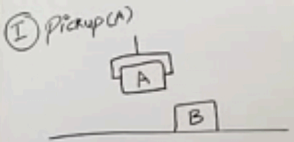


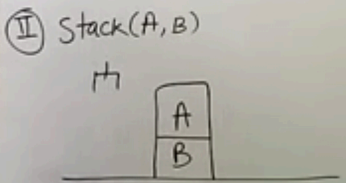


Goal Stack Planning









2. Semantic Network Representation

Semantic networks are alternative of predicate logic for knowledge representation. In Semantic networks, we can represent our knowledge in the form of graphical networks. This network consists of nodes representing objects and arcs which describe the relationship between those objects. Semantic networks can categorize the object in different forms and can also link those objects. Semantic networks are easy to understand and can be easily extended.

This representation consist of mainly two types of relations:

a. IS-A relation (Inheritance)
b. Kind-of-relation

Example: Following are some statements which we need to represent in the form of nodes and arcs.

Statements:

a. Jerry is a cat.

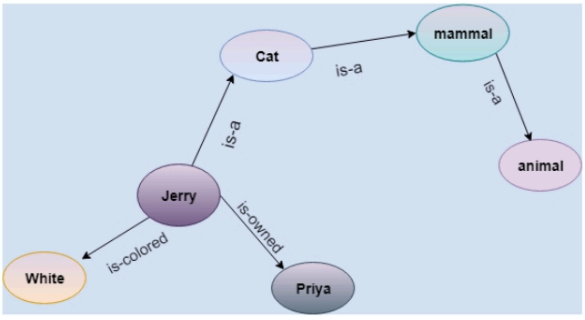
This representation consist of mainly two types of relations:

a. IS-A relation (Inheritance)
b. Kind-of-relation

Example: Following are some statements which we need to represent in the form of nodes and arcs.

Statements:

a. Jerry is a cat.
b. Jerry is a mammal
c. Jerry is owned by Priya.
d. Jerry is brown colored.
e. All Mammals are animal.



In the above diagram, we have represented the different type of knowledge in the form of nodes and arcs. Each object is connected with another object by some relation.

Drawbacks in Semantic representation:

1. Semantic networks take more computational time at runtime as we

Terms used in Reinforcement Learning

- **Agent():** An entity that can perceive/explore the environment and act upon it.
- **Environment():** A situation in which an agent is present or surrounded by. In RL, we assume the stochastic environment, which means it is random in nature.
- **Action():** Actions are the moves taken by an agent within the environment.
- **State():** State is a situation returned by the environment after each action taken by the agent.
- **Reward():** A feedback returned to the agent from the environment to evaluate the action of the agent.
- **Policy():** Policy is a strategy applied by the agent for the next action based on the current state.
- **Value():** It is expected long-term retuned with the discount factor and opposite to the short-term reward.
- **Q-value():** It is mostly similar to the value, but it takes one additional parameter as a current action (a).

Key Features of Reinforcement Learning

- In RL, the agent is not instructed about the environment and what actions need to be taken.

There are mainly three ways to implement reinforcement-learning in ML, which are:

1. **Value-based:**
   The value-based approach is about to find the optimal value function, which is the maximum value at a state under any policy. Therefore, the agent expects the long-term return at any state(s) under policy π.
2. **Policy-based:**
   Policy-based approach is to find the optimal policy for the maximum future rewards without using the value function. In this approach, the agent tries to apply such a policy that the action performed in each step helps to maximize the future reward.
   The policy-based approach has mainly two types of policy:
   - **Deterministic:** The same action is produced by the policy (π) at any state.
   - **Stochastic:** In this policy, probability determines the produced action.
3. **Model-based:** In the model-based approach, a virtual model is created for the environment, and the agent explores that environment to learn it. There is no particular solution or algorithm for this approach because the model representation is different for each environment.

Elements of Reinforcement Learning

There are four main elements of

Dictionary

Definitions from Oxford Languages
Learn more

English & Hindi

Search for a word

verb

व़ब्

GRAMMAR

noun

a word or group of words that is used to indicate that something happens or exists, for example bring, happen, be, do

क्रिया (ऐसा शब्द या वाक्यांश जो किसी घटना के घटने या किसी वस्तु के अस्तित्व में होने का अर्थ दे, जैसे 'bring', 'happen', 'be', 'do')

Feedback

Translations and more definitions

what is preposition

व्हाट इस प्रेपोजिशन

what is preposition with example

Dictionary

Definitions from Oxford Languages
Learn more

English & Hindi

Search for a word

preposition

ˌप्रेपəˈज़िश्न्

GRAMMAR

noun

a word or phrase that is used before a noun or pronoun to show place, time, direction, etc.

स्थान, समय, दिशा आदि के संकेत के लिए संज्ञा या सर्वनाम से पहले प्रयुक्त शब्द या वाक्यांश; पूर्वसर्ग

‘In’, ‘for’, ‘to’ and ‘out of’ are all prepositions.

Feedback

Translations and more definitions

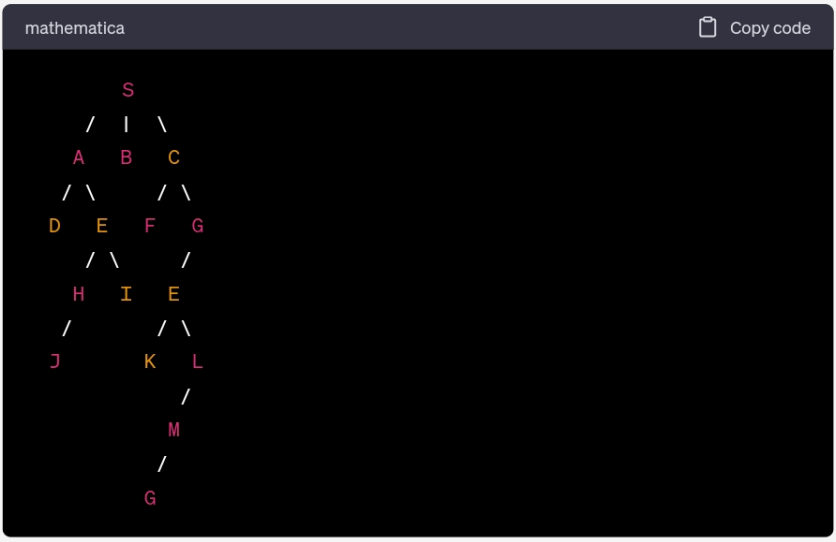
State space search is a fundamental concept in artificial intelligence (AI) used to solve problems by exploring the possible states of a problem domain. There are several types of state space search algorithms that differ in the way they explore and navigate through the problem space. Here are three common types:

1. Breadth-First Search (BFS):
   BFS explores all the neighboring states of a given state before moving on to the next level of states. It uses a queue data structure to maintain the order of exploration. BFS guarantees finding the shallowest goal state, but it may require a large amount of memory when dealing with large search spaces.

Example: Consider a maze where the goal is to find the shortest path from the start to the exit. In BFS, the algorithm would explore all the neighboring cells of the start cell before moving on to the next level of cells. It continues this process until it reaches the exit.

Diagram:

mathematica Copy code



2. Depth-First Search (DFS):
   DFS explores as far as possible along each branch before backtracking. It uses a stack data structure to maintain the order of exploration. DFS may find a goal state quickly but does not guarantee finding the shallowest goal state.

Example: Consider a problem where you are searching for a specific item in a tree structure. In DFS, the algorithm would start at the root node and explore the first child node, then its child nodes, and so on until it reaches a leaf node. If the item is not found, it backtracks and explores the next sibling node.

Diagram:

guarantee finding the shallowest goal state.

Example: Consider a problem where you are searching for a specific item in a tree structure. In DFS, the algorithm would start at the root node and explore the first child node, then its child nodes, and so on until it reaches a leaf node. If the item is not found, it backtracks and explores the next sibling node.

Diagram:

```mathematica
        A
      / | \
     B  D  F
    / \   /
   C   E G
```

3. A* Search:
   A* search combines the advantages of both BFS and heuristic-based search. It uses a heuristic function to estimate the cost from the current state to the goal and guides the search towards the most promising states. A* maintains a priority queue based on the cost function, which is a combination of the path cost and the heuristic estimate.

Example: Consider a navigation problem where you want to find the shortest path from one location to another on a map. A* search would use the estimated distance (heuristic) between the current location and the goal location to prioritize exploring states that are likely to lead to the shortest path.

Diagram: Not applicable in this case, as the diagram depends on the specific problem domain and the heuristics used.

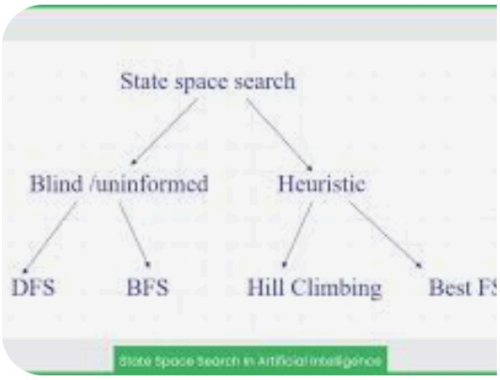
These are just a few examples of state space search algorithms in AI. The choice of algorithm depends on the problem at hand, the characteristics of the search space, and the available resources.

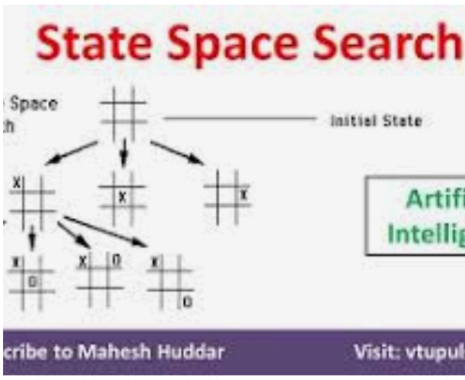
Google

types of state space s...

Images | Videos | Geeksforgeek | Ppt | E

हिंदी में खोजें | टाइप्स ऑफ़ स्टेट स्पेस सर्च इन ...





Examples of state-space search algorithms

- Traditional depth-first search.
- Breadth-first search.
- Iterative deepening.
- Lowest-cost-first search / Uniform-cost search (UCS)

w https://en.m.wikipedia.org › wiki

State space search - Wikipedia

About featured snippets | Feedback

Partially ordered plans are a type of planning representation used in artificial intelligence to model actions and their dependencies in a non-sequential manner. In a partially ordered plan, actions are partially ordered based on their dependencies, allowing for parallel and flexible execution of actions. This representation allows for more efficient planning by exploiting concurrency and reducing unnecessary ordering constraints.

In a partially ordered plan, actions are represented as nodes, and the dependencies between actions are represented as directed edges. These dependencies indicate that one action must be executed before or after another. However, there can be multiple actions that are not directly dependent on each other and can be executed in parallel.

Here's an example to illustrate partially ordered plans:

Consider a cooking scenario where you need to prepare a salad and a pasta dish. The actions involved are:

1. Chop vegetables
2. Boil water
3. Cook pasta
4. Mix salad dressing
5. Toss salad

In a sequential plan, you would perform these actions in a specific order. However, in a partially ordered plan, you can represent the dependencies and allow for parallel execution. Let's represent the dependencies using a directed graph:

markdown Copy code

```
          Boil water
             |
          Cook pasta
         /        \
Chop vegetables  Mix salad dressing
         \        /
          Toss salad
```


In this example, "Boil water" and "Cook pasta" are dependent on each

3. Cook pasta
4. Mix salad dressing
5. Toss salad

In a sequential plan, you would perform these actions in a specific order. However, in a partially ordered plan, you can represent the dependencies and allow for parallel execution. Let's represent the dependencies using a directed graph:

markdown Copy code

```
          Boil water
             |
          Cook pasta
         /        \
Chop vegetables  Mix salad dressing
         \        /
          Toss salad
```

In this example, "Boil water" and "Cook pasta" are dependent on each other, as pasta cannot be cooked without boiling water. However, "Chop vegetables" and "Mix salad dressing" are independent of each other and can be executed in parallel. Finally, "Toss salad" depends on both the completion of chopping vegetables and mixing salad dressing.

By representing the plan in a partially ordered manner, we can exploit parallelism and execute actions simultaneously. For example, while waiting for the water to boil, we can chop vegetables and mix the salad dressing concurrently.

Partially ordered plans provide more flexibility and efficiency in planning, especially when there are actions that can be executed independently or in parallel. They allow for concurrent execution and reduce unnecessary ordering constraints, leading to more efficient plans.

Quiz 6. Naïve Bayes classifiers. Again

Consider the following dataset

N	Color	Type	Origin	Stolen?
1	red	sports	domestic	yes
2	red	sports	domestic	no
3	red	sports	domestic	yes
4	yellow	sports	domestic	no
5	yellow	sports	imported	yes
6	yellow	SUV	imported	no
7	yellow	SUV	imported	yes
8	yellow	SUV	domestic	no
9	red	SUV	imported	no
10	red	sports	domestic	yes

Classify **(red, SUV, domestic)** using Naïve Bayes classifier

For this you need:

P(Stolen=yes)=1/2

P(Stolen=no)=1/2

P(red|Stolen=yes)=3/5

P(red|Stolen=no)=2/5

P(SUV|Stolen=yes)=1/5

P(SUV|Stolen=no)=3/5

P(domestic|Stolen=yes)=3/5

P(domestic|Stolen=no)=3/5

Do NOT apply Laplace correction, and do NOT compute actual probabilities. Just give the odds for this car to be stolen vs. safe.

P(Stolen=yes | red, SUV, domestic)=α P(red|Stolen=yes)
P(SUV|Stolen=yes)P(domestic|Stolen=yes) P(Stolen=yes)= α * 3/5*1/5*3/5*1/2

1

= α * 9/250

P(Stolen=no | red, SUV, domestic)=α P(red|Stolen=no)
P(SUV|Stolen=no)P(domestic|Stolen=no) P(Stolen=no)= α * 2/5*3/5*3/5*1/2

= α * 18/250

Odds 2:1 that this car is safe (will NOT be stolen)

Each **internal node** of the tree corresponds to an attribute.

Each **leaf node** corresponds to a Class Label.

In the decision tree for predicting a class label for a record, we start from the root of the tree. We compare the value of the root attribute with the record's attribute on the basis of comparison. We follow the branch corresponding to that value & jump to the next node. We continue comparing our record's attribute value with other internal nodes of the tree until we reach a leaf node.

1.1 Information Gain:

$$\text{Information Gain} = I(p, n)$$

$$I(p, n) = \frac{-p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log_2\left(\frac{n}{p+n}\right)$$

Here what are P and n? So to find p and n we check our class attribute or outcome which binary (0, 1). So for p, we take true value 1 (in case of binary) and for now, we take the false value 0 (binary value). We go deeper into the mathematical part just here introduction.

1.2 Entropy:

Entropy is basically used to create a tree. We find our entropy from attribute or class.

$$\text{Entropy } E(A) = \sum_{i=1}^{v} \frac{p_i+n_i}{p+n}(I(p,n))$$

1.3 Gain:

$$\text{Gain} = \text{Information Gain} - \text{Entropy}$$

$$\text{Gain} = I(p, n) - E(A)$$

Gain is basically used to find one by one attribute of our training set.

2. How Does Decision Tree work in Classification and Regression?

2.1 Classification:

This dataset has 10 instances and four numbers of attributes. Here first three attributes are predictor and the last attribute is the target attribute.

Age	Competition	Type	Profit
old	yes	software	down