CS3243: Introduction to Artificial Intelligence
Lecture Notes 2: Uninformed Search

---

1. Types of Agents

Agent Type	Activity	State Depends On
Reflex	Passive	Percepts
Model-Based	Passive	Percepts, Model of the world
Goal-Based	Active	Percepts, Model of the world, Current state
Utility-Based	Active	May have multiple goals that may be inconsistent with each other, and it state also depends on its utility function.
Learning	Active	Can update its agent program based on its experiences.

2. Mopbot as an Example of a Goal-Based Agent

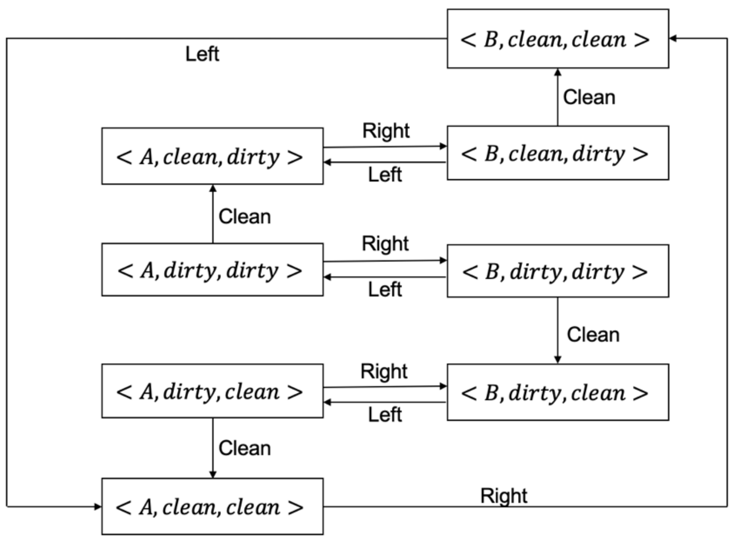
Let us model the behavior of a Mopbot in a two-room apartment. This Mopbot is a goal-based agent in a deterministic, static and fully observable task environment. We shall define the following useful abstractions for this agent:

- Every **State** which Mopbot can take is of the form $< loc, status(A), status(B) >$, where loc refers to the location of the Mopbot, and the status of a room can either be $Dirty$ or $Clean$.

- The set of **Actions** in which Mopbot can take is $\{Left, Right, Clean, Idle\}$.

- The **Performance Measure** of Mopbot is defined as a function $h: State \times Action \to \mathbb{R}$.

- The **Transition Model** $g: State \times Actions \to State$ gives the next state that Mopbot lands on given a current state and a sequence of actions. It can be modelled as a graph.



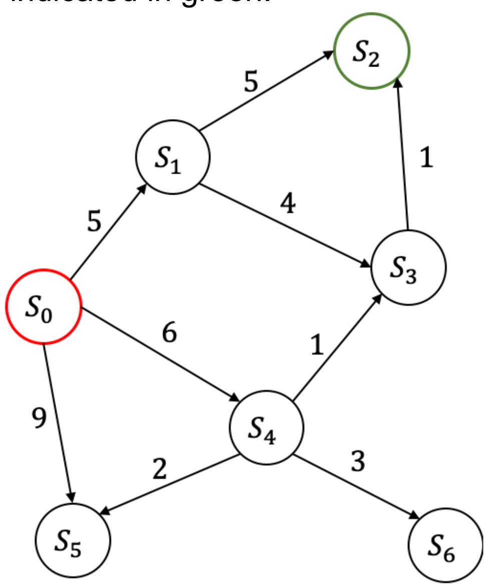
Note: A transition model is not the same as an **agent function**. The latter is a function that takes in a State and a set of percepts, and outputs the next action in which an agent should take.

3. The Transition Model as a Graph

The graphical representation of a transition model can be very large. Within the graph, the following parameters are of importance:

- **Branching factor (b):** The maximum number of edges out of a state.
- **Depth (d):** The minimum path length from the start state to the goal state.
- **Maximum (m):** The maximum length of a path from the start state to any state in the graph.
- **Performance Measure:** The minimum path cost from the start state to a goal state.

Consider the following transition model below. The start state S_0 is indicated in red, while the goal state S_2 is indicated in green.



In the above data structure, each **node** of the graph should contain the (1) state of the agent, (2) parent node, (3) action to generate the node, and the (4) path cost to reach that node.

From a start node, one can perform a **Tree-Search** or a **Graph-Search**. The latter is faster, but requires more space as we have to keep track of nodes that have already been visited.

4. Breadth-First Search (BFS)

Given an initial node u, we want to find the path from u to a goal node within a graph using the following algorithm.

```
FindPathToGoal(u):
    if GoalTest(u):
        return path(u)
    Frontier ← Queue(u)
    Explored ← {u}
    while Frontier is not empty:
        u ← Frontier.pop()
        for all children v of u:
            if (GoalTest(v)):
                return path(v)
            else:
                if (v not in Explored):
                    Explored.add(v)
                    Frontier.push(v)
    return Failure
```

We will have to consider the following in order to evaluate an algorithm:

(a) Completeness: The algorithm will find a path to the goal if the goal is reachable from the start state.

Breadth-First Search is complete.

Sketch of Proof:
Assume that there is a path π of length $(k+1)$ from start state S_0 to the goal state S_g.

$$\pi: S_0, S_{i_1}, S_{i_2}, \cdots, S_{i_k}, S_g$$

Base Case:
$(k+1) = 0$ and $S_0 = S_g$. In this case, the start state is already the goal state, and there indeed exists a path.

Inductive Hypothesis:
Let us assume that the agent has reached nodes at a distance $\leq k$ steps from the start node.

Hence, at the current node S_{i_k}, either

- S_g has already been explored, because π might not be a shortest path to S_g and there exists another shorter path to S_g. Or,
- S_g will be explored, because S_g is a child of S_{i_k}.

(b) Optimality: When the algorithm finds a path, the path will be of minimum cost.

The algorithm given above is not optimal.

(c) Time: The time complexity of this algorithm is $b + b^2 + \cdots + b^d = O(b^{d+1})$.

(d) Space: The space complexity of the set of Explored nodes is $O(b^{d+1})$, and that of the frontier is $O(b^d)$.